

Axiomatic theory of hereditarily finite sets and its fragments.

Štěpán Holub Zuzana Haniková

July 14, 2026

Funded by the Czech Science Foundation grant GAČR 25-16489S.

Abstract

Axiomatization of the theory of hereditarily finite sets in classical first-order logic is systematically explored and formalized. Our list of axioms includes the usual ones of the Zermelo-Fraenkel set theory, as well as some of their less usual variants and alternatives. The strongest theory considered is ZFfin , obtained from ZF by replacing the axiom of infinity with its negation and adding the axiom of transitive closure. Of particular interest are the axioms used in the set fragment of the Alternative Set Theory by Vopěnka [10, 7, 8, 9], which provide a different approach to axiomatizing ZFfin . The hierarchy of fragments of ZFfin is represented as a hierarchy of locales.

Set-theoretic membership, the only basic predicate in the language of the theory, is rendered as a polymorphic binary predicate. Accordingly, first-order formulas of this language are understood as predicates over that type. ZFfin is not finitely axiomatizable, so it is necessary to use axiom schemata. To that end, an inductive definition of set-theoretically definable predicates is introduced and their basic properties are elaborated.

A major aim of the formalization is to study several axioms of finiteness and to formalize the equivalence of the fragments in which these axioms are used. In particular, we formalize the equivalence of ZFfin and the set fragment of Vopěnka's Alternative Set Theory, where the latter uses the schema of induction for sets as a finiteness principle. Next to that, Tarski finiteness and Dedekind finiteness are used. The formalization also focuses on the axioms of transitive closure, regularity, and the axiom schema of epsilon induction and their interplay in the context of various fragments.

Hereditarily finite sets, previously formalized by Paulson, satisfy the theory ZFfin . To demonstrate independence of some statements, several non-standard models of specific fragments are constructed. We formalize the Bernays-Rieger method of permutation models, and using the construction of a model dating back to [6] we formalize the proof that neither the scheme of regularity nor the existence of a transitive closure follow from regularity for finite sets.

This formalization takes inspiration from various sources but does not follow closely any of them. In addition to already cited literature, it covers some results from [1], [2], [4] and [3].

Contents

1	Fragments of Zermelo-Fraenkel set theory without infinity	2
1.1	Preliminaries	2
1.2	Signature	2
1.2.1	Membership and basic set-theoretic notions	2
1.2.2	Set theoretical notions	3
1.2.3	Set relations and properties	7
1.3	Axiomatizations of hereditarily finite sets	9
1.3.1	Finiteness axioms	9
1.3.2	Extensionality	10
1.3.3	Empty set	11
1.3.4	Set pair	12
1.3.5	Set successor	13
1.3.6	Regularity	16
1.3.7	Separation	18
1.3.8	Transitive superset	21
1.3.9	Replacement	22
1.3.10	Powerset	23
1.3.11	Union	24
1.3.12	Successor induction	30
1.3.13	Negation of inf	31
1.3.14	Dedekind finite	32
1.3.15	Tarski finite	32
1.3.16	Set induction and regularity schema	33
1.3.17	Summary of dependencies	34
2	Models and counter-models	37
2.1	Hereditarily finite sets	37
2.2	Permutation models	37
2.3	Regularity	38
2.4	Independence of transitive superset	40
	References	44

Chapter 1

Fragments of Zermelo-Fraenkel set theory without infinity

```
theory ZFfin
imports Main
begin
```

We explore axiomatizations of the theory of hereditarily finite sets, their fragments and relationships between them.

1.1 Preliminaries

```
lemma ex-or-iff-or [simp]:  $(\exists w. (w = x \vee w = y) \wedge P\ u\ w) \longleftrightarrow (P\ u\ x \vee P\ u\ y)$ 
  <proof>
lemma abstract-foundation-iff:  $((\exists x. P\ x) \longrightarrow (\exists x. B\ x \wedge P\ x)) \longleftrightarrow (\forall x. B\ x \longrightarrow \neg P\ x) \longrightarrow (\forall x. \neg P\ x)$ 
  <proof>
```

1.2 Signature

1.2.1 Membership and basic set-theoretic notions

```
locale set-signature =
  fixes membership :: 'a  $\Rightarrow$  'a  $\Rightarrow$  bool (infix  $\varepsilon$  50)
```

```
begin
```

General relation between regularity schema and epsilon-induction scheme.
Applying the above remark about B-induction and B-foundation

```
thm abstract-foundation-iff[of  $\lambda x. \neg P\ x \lambda x. (\forall y. y \varepsilon x \longrightarrow (P\ y))$ , unfolded  
not-not, symmetric]
```

thm *abstract-foundation-iff*[of $\lambda y. P y \lambda x. (\forall y. y \in x \longrightarrow \neg (P y))$, *unfolded not-not*]

1.2.2 Set theoretical notions

We use "M" (as in "Menge") for our defined versions of set-theoretical concepts, built upon membership.

definition *subsetM* :: 'a $\Rightarrow$ 'a $\Rightarrow$ bool (- $\subseteq_M$ - [51,51] 53) **where**
subsetM x y $\equiv (\forall z. z \in x \longrightarrow z \in y)$

lemma *subsetM-refl[simp]*: $x \subseteq_M x$
<proof>

lemma *subsetM-trans[simp]*: **assumes** $x \subseteq_M y \ y \subseteq_M z$ **shows** $x \subseteq_M z$
<proof>

definition *proper-subsetM* :: 'a $\Rightarrow$ 'a $\Rightarrow$ bool (- $\subset_M$ - [50,50] 50) **where**
proper-subsetM x y $\equiv (\forall z::'a. z \in x \longrightarrow z \in y) \wedge x \neq y$

named-theorems *set-defs*

lemmas[*set-defs*] = *proper-subsetM-def subsetM-def*

lemma *proper-subset-def'*: $x \subset_M y \longleftrightarrow x \subseteq_M y \wedge x \neq y$
<proof>

lemma *proper-subsetI*: $x \subseteq_M y \Longrightarrow x \neq y \Longrightarrow x \subset_M y$
<proof>

lemma *proper-subsetM-irrefl[simp]*: $\neg x \subset_M x$
<proof>

definition *transM*:: 'a $\Rightarrow$ bool **where**
transM x $\equiv \forall y. y \in x \longrightarrow y \subseteq_M x$

Hilbert's description operator *The* is used to define sets that are determined by their elements. Resulting set-theoretic operations can and will be used in appropriate contexts only that guarantee the existence of corresponding sets, and their unicity (provable with extensionality).

definition *collectM* :: ('a $\Rightarrow$ bool) $\Rightarrow$ 'a **where**
collectM Q $\equiv THE z. (\forall u. (u \in z \longleftrightarrow Q u))$

definition *emptysetM* ($\emptyset$) **where**
emptysetM $\equiv collectM (\lambda x. x \neq x)$

definition *separationM* :: 'a $\Rightarrow$ ('a $\Rightarrow$ bool) $\Rightarrow$ 'a **where**
separationM y Q $\equiv collectM (\lambda u. u \in y \wedge Q u)$

definition *powersetM* :: 'a $\Rightarrow$ 'a ($\mathfrak{P}$) **where**

$\text{powersetM } x \equiv \text{collectM } (\lambda u. u \subseteq_M x)$

definition $\text{binunionM} :: 'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\cup_M$ 55) **where**
 $\text{binunionM } x y \equiv \text{collectM } (\lambda u. u \varepsilon x \vee u \varepsilon y)$

definition $\text{intersectionM} :: 'a \Rightarrow 'a \Rightarrow 'a$ ($- \cap_M -$ [55,54] 60) **where**
 $\text{intersectionM } x y \equiv \text{collectM } (\lambda u. u \varepsilon x \wedge u \varepsilon y)$

lemma *intersection-symm*: $x \cap_M y = y \cap_M x$
 $\langle \text{proof} \rangle$

definition $\text{unionM} :: 'a \Rightarrow 'a$ ($\bigcup_M$) **where**
 $\text{unionM } x \equiv \text{collectM } (\lambda u. (\exists v. v \varepsilon x \wedge u \varepsilon v))$

definition $\text{differenceM} :: 'a \Rightarrow 'a \Rightarrow 'a$ ($- \setminus_M -$) **where**
 $\text{differenceM } x y \equiv \text{collectM } (\lambda u. u \varepsilon x \wedge \neg u \varepsilon y)$

definition $\text{singletonM} :: 'a \Rightarrow 'a$ ($\{-\}_M$ 70) **where**
 $\text{singletonM } x \equiv \text{collectM } (\lambda u. u = x)$

definition $\text{pairM} :: 'a \Rightarrow 'a \Rightarrow 'a$ ($\{-,-\}_M$ [51,51] 60) **where**
 $\text{pairM } x y \equiv \text{collectM } (\lambda u. u = x \vee u = y)$

— Set successor. Also known as adjunction.

definition $\text{setsucM} :: 'a \Rightarrow 'a \Rightarrow 'a$ (**suc**) **where**
 $\text{setsucM } x y \equiv \text{collectM } (\lambda u. u \varepsilon x \vee u = y)$

definition $\text{replaceM} :: ('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \Rightarrow 'a$ **where**
 $\text{replaceM } P x \equiv \text{collectM } (\lambda u. \exists v. v \varepsilon x \wedge P v u)$

definition $\text{leastM} :: ('a \Rightarrow \text{bool}) \Rightarrow 'a$ ($\bigcap_M$) **where**
 $\text{leastM } Q \equiv \text{collectM } (\lambda u. \forall y. Q y \longrightarrow u \varepsilon y)$

Least transitive superset

definition $\text{least-tsM} :: 'a \Rightarrow 'a$ **where**
 $\text{least-tsM } x = \bigcap_M (\lambda y. \text{transM } y \wedge x \subseteq_M y)$

definition $\text{ordered-pairM} :: 'a \Rightarrow 'a \Rightarrow 'a$ ($\langle -, - \rangle$ [51,51] 60) **where**
 $\text{ordered-pairM } a b \equiv \{\{a\}_M, \{a, b\}_M\}_M$

definition $\text{relationM} :: 'a \Rightarrow \text{bool}$ **where**
 $\text{relationM } x \equiv \forall v. v \varepsilon x \longrightarrow (\exists a b. v = \langle a, b \rangle)$

definition $\text{rel-inverseM} :: 'a \Rightarrow 'a$ **where**
 $\text{rel-inverseM } r \equiv \text{collectM } (\lambda u. \exists a b. \langle a, b \rangle \varepsilon r \wedge u = \langle b, a \rangle)$

definition $\text{functionM} :: 'a \Rightarrow \text{bool}$ **where**
 $\text{functionM } x \equiv \text{relationM } x \wedge (\forall a b c. \langle a, b \rangle \varepsilon x \wedge \langle a, c \rangle \varepsilon x \longrightarrow b = c)$

lemma *funD*: $\text{functionM } f \implies \langle a, b \rangle \varepsilon f \implies \langle a, c \rangle \varepsilon f \implies b = c$
 $\langle \text{proof} \rangle$

definition *domM* :: $'a \Rightarrow 'a$ **where**
 $\text{domM } r \equiv \text{collectM } (\lambda u. \exists v. \langle u, v \rangle \varepsilon r)$

definition *rngM* :: $'a \Rightarrow 'a$ **where**
 $\text{rngM } r \equiv \text{collectM } (\lambda u. \exists v. \langle v, u \rangle \varepsilon r)$

definition *one-oneM* :: $'a \Rightarrow \text{bool}$ **where**
 $\text{one-oneM } f \equiv \text{functionM } f \wedge (\forall a \ b \ c. \langle b, a \rangle \varepsilon f \wedge \langle c, a \rangle \varepsilon f \longrightarrow b = c)$

lemma *one-one-inj*: $\text{one-oneM } f \implies \langle b, a \rangle \varepsilon f \implies \langle c, a \rangle \varepsilon f \implies b = c$
 $\langle \text{proof} \rangle$

lemma *one-oneI*: **assumes** $\forall v. v \varepsilon f \longrightarrow (\exists a \ b. v = \langle a, b \rangle) (\forall a \ b \ c. \langle b, a \rangle \varepsilon f \wedge \langle c, a \rangle \varepsilon f \longrightarrow b = c)$
shows $\text{one-oneM } f$
 $\langle \text{proof} \rangle$

lemma *one-oneD1*: $\text{one-oneM } f \implies \forall v. v \varepsilon f \longrightarrow (\exists a \ b. v = \langle a, b \rangle)$
 $\langle \text{proof} \rangle$

lemma *one-oneD2*: $\text{one-oneM } f \implies (\forall a \ b \ c. \langle b, a \rangle \varepsilon f \wedge \langle c, a \rangle \varepsilon f \longrightarrow b = c)$
 $\langle \text{proof} \rangle$

lemma *one-oneD3*: $\text{one-oneM } f \implies (\forall a \ b \ c. \langle a, b \rangle \varepsilon f \wedge \langle a, c \rangle \varepsilon f \longrightarrow b = c)$
 $\langle \text{proof} \rangle$

definition *set-equivalent* :: $'a \Rightarrow 'a \Rightarrow \text{bool}$ ($- \approx_M -$ [51,51] 52) **where**
 $\text{set-equivalent } x \ y \equiv (\exists f. \text{one-oneM } f \wedge x = \text{domM } f \wedge y = \text{rngM } f)$

definition *cartesian-productM* :: $'a \Rightarrow 'a \Rightarrow 'a$ ($- \times_M -$ [51,51] 60) **where**
 $\text{cartesian-productM } x \ y \equiv \text{collectM } (\lambda u. \exists v_1 \ v_2. v_1 \varepsilon x \wedge v_2 \varepsilon y \wedge u = \langle v_1, v_2 \rangle)$

definition *composeM* :: $'a \Rightarrow 'a \Rightarrow 'a$ ($- \circ_M -$ [51,51] 60) **where**
 $f \circ_M g = \text{collectM } (\lambda u. \exists a \ b \ c. \langle a, b \rangle \varepsilon g \wedge \langle b, c \rangle \varepsilon f \wedge u = \langle a, c \rangle)$

definition *tarski-fin* :: $'a \Rightarrow \text{bool}$ **where**
 $\text{tarski-fin } x \equiv \forall y. (\forall z. z \varepsilon y \longrightarrow z \subseteq_M x) \longrightarrow (\exists z. z \varepsilon y) \longrightarrow (\exists z. z \varepsilon y \wedge \neg(\exists w. w \varepsilon y \wedge z \subset_M w))$

lemma *tarski-subset-tarski*: **assumes** $\text{tarski-fin } x \ y \subseteq_M x$ **shows** $\text{tarski-fin } y$
 $\langle \text{proof} \rangle$

definition *dedekind-fin* :: $'a \Rightarrow \text{bool}$ **where**
 $\text{dedekind-fin } x \equiv \forall y. (y \subset_M x \longrightarrow \neg x \approx_M y)$

— x is regular if it contains no infinite ε -decreasing chain as a subset

definition *regular* :: 'a $\Rightarrow$ bool **where**

regular $x \equiv \forall y. y \subseteq_M x \longrightarrow (\exists z. z \varepsilon y) \longrightarrow (\exists z. z \varepsilon y \wedge (\forall v. \neg (v \varepsilon z \wedge v \varepsilon y)))$

— Ordinals and natural numbers

definition *epschain* :: 'a $\Rightarrow$ bool **where**

epschain $x \equiv \text{transM } x \wedge (\forall y z. y \varepsilon x \wedge z \varepsilon x \longrightarrow y \varepsilon z \vee y = z \vee z \varepsilon y)$

— In theories that do not rely on the regularity axiom, ordinals are explicitly assumed not to contain infinite ε -decreasing chains

definition *ordM* :: 'a $\Rightarrow$ bool **where**

ordM $x \equiv \text{epschain } x \wedge \text{regular } x$

lemma *ordM-I*: *regular* $x \Longrightarrow \text{epschain } x \Longrightarrow \text{ordM } x$

<proof>

lemma *ordM-regular[simp]*: *ordM* $x \Longrightarrow \text{regular } x$

<proof>

lemma *ordM-D1*: *ordM* $x \Longrightarrow y \varepsilon x \Longrightarrow y \subseteq_M x$

<proof>

lemma *ordM-trans*: **assumes** *ordM* v $w \varepsilon u$ $u \varepsilon v$ **shows** $w \varepsilon v$

<proof>

lemma *ordM-D2*: *ordM* $x \Longrightarrow y \varepsilon x \Longrightarrow z \varepsilon x \Longrightarrow y \varepsilon z \vee y = z \vee z \varepsilon y$

<proof>

definition *is-sucM* :: 'a $\Rightarrow$ bool **where**

is-sucM $x \equiv \exists y. (\forall z. z \varepsilon x \longleftrightarrow z \varepsilon y \vee z = y)$

— A natural number is an ordinal x such that: x and all its elements are either empty or a successor

definition *natM* :: 'a $\Rightarrow$ bool **where**

natM $x \equiv \text{ordM } x \wedge (\forall v. (v \varepsilon x \vee v = x) \longrightarrow (\exists u. u \varepsilon v) \longrightarrow \text{is-sucM } v)$

lemma *natM-I*: *ordM* $x \Longrightarrow \text{is-sucM } x \Longrightarrow (\bigwedge v. \exists u. u \varepsilon v \Longrightarrow v \varepsilon x \Longrightarrow \text{is-sucM } v) \Longrightarrow \text{natM } x$

<proof>

lemma *nat-is-suc*: *natM* $x \Longrightarrow \exists u. u \varepsilon x \Longrightarrow \text{is-sucM } x$

<proof>

lemma *nat-mem-is-suc*: *natM* $x \Longrightarrow v \varepsilon x \Longrightarrow \exists u. u \varepsilon v \Longrightarrow \text{is-sucM } v$

<proof>

lemma *natM-D*: *natM* $x \Longrightarrow \text{ordM } x$

<proof>

definition *cardinality* :: 'a $\Rightarrow$ 'a $\Rightarrow$ bool **where**

cardinality x n $\equiv$ natM n $\wedge$ x $\approx_M$ n

lemmas[*set-defs*] = *transM-def epschain-def ordM-def tarski-fin-def is-sucM-def*
natM-def functionM-def relationM-def one-oneM-def rngM-def domM-def regular-def
cardinality-def set-equivalent-def

1.2.3 Set relations and properties

We represent a set formula as the predicate it defines. A predicate assigns truth values to assignments. We consider countably many variables x_i , represented by natural numbers. An assignment maps variables to elements of the domain, i.e., it maps *nat* to 'a. Predicates defined by set formulas are defined inductively.

inductive *SetFormulaPredicate* :: ((nat $\Rightarrow$ 'a) $\Rightarrow$ bool) $\Rightarrow$ bool

where

SFP-mem: $\bigwedge m n. \text{SetFormulaPredicate } (\lambda \Xi. (\Xi m) \varepsilon (\Xi n))$ — formula $x_m \varepsilon x_n$
| *SFP-eq*: $\bigwedge m n. \text{SetFormulaPredicate } (\lambda \Xi. (\Xi m) = (\Xi n))$ — formula $x_m = x_n$
| *SFP-neg*: *SetFormulaPredicate* P $\Longrightarrow$ *SetFormulaPredicate* ($\lambda \Xi. \neg P \Xi$) — formula $\neg \varphi$
| *SFP-disj*: *SetFormulaPredicate* P $\Longrightarrow$ *SetFormulaPredicate* Q
 $\Longrightarrow$ *SetFormulaPredicate* ($\lambda \Xi. P \Xi \vee Q \Xi$) — formula $\varphi \vee \psi$
| *SFP-all*: $\bigwedge n. \text{SetFormulaPredicate } P \Longrightarrow \text{SetFormulaPredicate } (\lambda \Xi. \forall a. P (\Xi(n:=a)))$
— formula $\forall x_n. \varphi$

named-theorems *SFP-rules*

lemmas[*SFP-rules*] = *SFP-mem SFP-eq SFP-neg SFP-disj SFP-all*

Every set-theoretically definable predicate depends on finitely many values. Such values correspond to free variables.

lemma *bounded-free*: **assumes** *SetFormulaPredicate* P

shows $\exists m. \forall \Xi \Xi'. (\forall i < m. \Xi i = \Xi' i) \longrightarrow P(\Xi) = P(\Xi')$

<proof>

lemma *transform-variables*: **assumes** *SetFormulaPredicate* P

shows *SetFormulaPredicate* ($\lambda \Xi. P (\lambda n. \Xi(f n))$)

<proof>

lemma *update-variable[intro]*: **assumes** *SetFormulaPredicate* P

shows *SetFormulaPredicate* ($\lambda \Xi. P(\Xi(n:= \Xi m))$)

<proof>

lemma *swap-variables*: **assumes** *SetFormulaPredicate* P

shows *SetFormulaPredicate* ($\lambda \Xi. P (\Xi(k := \Xi l, l := \Xi k))$)

<proof>

A rule allowing to get rid of quantifiers when proving that a predicate is a *SetFormulaPredicate*

lemma *sfp-all-rule4* [*SFP-rules*]: **assumes** $\bigwedge m. \text{SetFormulaPredicate } (\lambda \Xi. Q (\Xi m) (\Xi k1) (\Xi k2) (\Xi k3) (\Xi k4))$
shows $\text{SetFormulaPredicate } (\lambda \Xi. \forall y. Q y (\Xi k1) (\Xi k2) (\Xi k3) (\Xi k4))$
 $\langle \text{proof} \rangle$

named-theorems *logsimps*

lemmas[*logsimps*] = *not-not*

lemma *reduce-ex* [*logsimps*]: $(\lambda \Xi. (\exists z. Q z \Xi)) = (\lambda \Xi. \neg (\forall z. \neg (Q z \Xi)))$
 $\langle \text{proof} \rangle$

lemma *reduce-and* [*logsimps*]: $(\lambda \Xi. (P \Xi) \wedge (Q \Xi)) = (\lambda \Xi. (\neg ((\neg (P \Xi)) \vee (\neg (Q \Xi)))))$
 $\langle \text{proof} \rangle$

lemma *reduce-imp*[*logsimps*]: $(\lambda \Xi. (P \Xi) \longrightarrow (Q \Xi)) = (\lambda \Xi. (\neg (P \Xi) \vee (Q \Xi)))$
 $\langle \text{proof} \rangle$

lemma *reduce-iff*[*logsimps*]: $(\lambda \Xi. (P \Xi) \longleftrightarrow (Q \Xi)) = (\lambda \Xi. \neg (\neg (\neg P \Xi \vee Q \Xi) \vee \neg (\neg Q \Xi \vee P \Xi)))$
 $\langle \text{proof} \rangle$

lemma *SFP-const*[*intro,simp*]: $\text{SetFormulaPredicate } (\lambda \Xi. b)$
 $\langle \text{proof} \rangle$

lemma *SFP-ex*: **assumes** $\text{SetFormulaPredicate } P$ **shows** $\text{SetFormulaPredicate } (\lambda \Xi. (\exists z. P (\Xi (m := z))))$
 $\langle \text{proof} \rangle$

lemma *SFP-replace*: **assumes** $\text{SetFormulaPredicate } P$ **and** *fresh-m*: $\forall \Xi \Xi'. (\forall i < m. \Xi i = \Xi' i) \longrightarrow P \Xi = P \Xi'$
shows $\text{SetFormulaPredicate } (\lambda \Xi. \exists z. \forall v. (v \varepsilon z) = (\exists u. u \varepsilon \Xi m \wedge P (\Xi (0:=u, 1:=v))))$
 $\langle \text{proof} \rangle$

A (binary) relation is set-theoretically definable if the predicate it defines by assigning values to variables x_0 and x_1 is set-theoretically definable. The value of the relation therefore cannot depend on other variables than x_0 and x_1 . In other words, if a formula φ defining a set-theoretically definable relation contains a free variable x_k , $1 < k$, then it is equivalent to $\forall x_k. \varphi$

definition *SetRelation* :: $('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow \text{bool}$
where

$\text{SetRelation } P \equiv \text{SetFormulaPredicate } (\lambda \Xi. P (\Xi 0) (\Xi 1))$

A set-theoretically definable property is defined similarly.

definition *SetProperty* :: $('a \Rightarrow \text{bool}) \Rightarrow \text{bool}$
where

$\text{SetProperty } P \equiv \text{SetFormulaPredicate } (\lambda \Xi. P (\Xi 0))$

Auxiliary proofs for specific useful set-relations and set-properties

lemma *SR-neg[simp, intro]*: *SetRelation* $P \implies \text{SetRelation } (\lambda x y. \neg P x y)$
 $\langle \text{proof} \rangle$

lemma *SR-sym[simp, intro]*: **assumes** *SetRelation* P **shows** *SetRelation* $(\lambda x y. P y x)$
 $\langle \text{proof} \rangle$

lemma *SR-neq[simp]*: *SetRelation* $(\neq)$
 $\langle \text{proof} \rangle$

lemma *SP-const[simp]*: *SetProperty* $(\lambda x. b)$
 $\langle \text{proof} \rangle$

lemma *SP-neg[simp, intro]*: *SetProperty* $P \implies \text{SetProperty } (\lambda x. \neg P x)$
 $\langle \text{proof} \rangle$

lemma *SP-tarski[simp]*: *SetProperty* *tarski-fin*
 $\langle \text{proof} \rangle$

lemma *SP-setsuc[simp]*: *SetProperty* *is-sucM*
 $\langle \text{proof} \rangle$

lemma *SP-nat[simp]*: *SetProperty* *natM*
 $\langle \text{proof} \rangle$

lemma *empty-mem-ord'*: **assumes** *ordM* $x \exists z. z \in x$
shows $\exists \text{ emp. } (\forall u. \neg u \in \text{emp}) \wedge \text{emp} \in x$
 $\langle \text{proof} \rangle$

end

1.3 Axiomatizations of hereditarily finite sets

1.3.1 Finiteness axioms

The aim of each of the axioms is to guarantee that each element of the domain will be finite.

Usual axiom: there is no inductive set.

locale *L-fin* = *set-signature* +
assumes *fin*: $\neg (\exists x. \emptyset \in x \wedge (\forall y. y \in x \longrightarrow \text{setsucM } y y \in x))$

Induction schema for set formulas, a.k.a. set induction schema or just set induction. Not to be confused with epsilon induction.

locale *L-setind* = *set-signature* +
assumes *setind*: $\bigwedge P. \text{SetFormulaPredicate } P \implies$

$P(\Xi(0:= \emptyset)) \longrightarrow (\forall x y. P(\Xi(0:= x)) \longrightarrow P(\Xi(0:= \text{setsucM } x y))) \longrightarrow (\forall x. P(\Xi(0:= x)))$

begin

lemma *setind-SP*: **assumes** *SetProperty P* **and** $P \emptyset$ **and** *step*: $\forall x y. P x \longrightarrow P(\text{setsucM } x y)$

shows $P x$

$\langle \text{proof} \rangle$

lemma *setind-var*: **assumes** *SetFormulaPredicate P* $P(\Xi(n:= \emptyset))$ **and** *step*: $\forall x y. P(\Xi(n:= x)) \longrightarrow P(\Xi(n:= \text{setsucM } x y))$

shows $P(\Xi(n:= x))$

$\langle \text{proof} \rangle$

end

locale *L-dedekind* = *set-signature* +

assumes *dedekind*: $\forall x. \text{dedekind-fin } x$

locale *L-tarski* = *set-signature* +

assumes *tarski*: $\forall y. \text{tarski-fin } y$

1.3.2 Extensionality

locale *L-setext* = *set-signature* +

assumes *setext*: $x = y \longleftrightarrow (\forall z. z \in x \longleftrightarrow z \in y)$

begin

set in HOL is defined as a class. Therefore, the following comprehension is an axiom.

lemma $x \in \{u. P u\} \longleftrightarrow P x$

$\langle \text{proof} \rangle$

In case of (our) M-sets, only uniqueness, not existence is guaranteed by extensionality.

lemma *collect-def'*:

assumes $\forall u. (u \in w \longleftrightarrow Q u)$

shows $\text{collectM } Q = w$

$\langle \text{proof} \rangle$

lemma *collect-def-ex*: — A formulation useful in proofs

assumes $\exists w. \forall u. (u \in w \longleftrightarrow Q u)$

shows $u \in \text{collectM } Q \longleftrightarrow Q u$

$\langle \text{proof} \rangle$

lemma *proper-subset-diff[simp]* : $y \subset_M x \implies \exists z. z \in x \wedge \neg z \in y$

$\langle \text{proof} \rangle$

lemma *subsetM-antisym*: $y \subseteq_M x \implies x \subseteq_M y \implies x = y$
 $\langle proof \rangle$

lemma *proper-subsetM-trans[simp]*: **assumes** $x \subset_M y$ $y \subset_M z$ **shows** $x \subset_M z$
 $\langle proof \rangle$

lemma *emptyset-mem-ord*: **assumes** $ordM\ x$ $\exists\ z. z \in x$ **shows** $\emptyset \in x$
 $\langle proof \rangle$

end

1.3.3 Empty set

locale *L-empty* = *set-signature* +
assumes *empty*: $\exists\ x. \forall\ y. (\neg y \in x)$

Since the empty set is defined *collectM*, and hence Hilbert's *The*, nothing useful can be said about $\emptyset$ without extensionality.

locale *L-setext-empty* = *L-setext* + *L-empty*

begin

— Here we can already prove that $\emptyset$, has the intended meaning.

lemma *empty-is-empty[simp]*: $\forall\ y. (\neg y \in \emptyset)$
 $\langle proof \rangle$

lemma *emptyset-def'[set-defs]*: $x = \emptyset \longleftrightarrow (\forall\ u. \neg u \in x)$
 $\langle proof \rangle$

lemma *empty-mem-false [set-defs]*: $u \in \emptyset \longleftrightarrow False$
 $\langle proof \rangle$

lemma *setsuc-empty-sing*: $setsucM\ \emptyset\ x = \{x\}_M$
 $\langle proof \rangle$

lemma *SFP-empty-n [simp]*: *SetFormulaPredicate* $(\lambda\Xi. \Xi\ n = \emptyset)$
 $\langle proof \rangle$

lemma *SP-empty[simp]*: *SetProperty* $(\lambda x. x = \emptyset)$
 $\langle proof \rangle$

lemma *SFP-empty-n' [simp]*: *SetFormulaPredicate* $(\lambda\Xi. \forall u. \neg u \in \Xi\ n)$
 $\langle proof \rangle$

lemma *empty-dedekind[simp]*: **shows** *dedekind-fin* $\emptyset$
 $\langle proof \rangle$

lemma *subset-of-empty[simp]*: $u \subseteq_M \emptyset \longleftrightarrow u = \emptyset$
 $\langle proof \rangle$

lemma *empty-tarski*[simp]: **shows** *tarski-fin* $\emptyset$
 ⟨proof⟩

lemma *nemp-setI*[intro]: $x \in y \implies y \neq \emptyset$
 ⟨proof⟩

lemma *nemp-setI-ex*: $\exists x. x \in y \implies y \neq \emptyset$
 ⟨proof⟩

lemma *empty-is-subset*: $\emptyset \subseteq_M A$
 ⟨proof⟩

lemma *empty-regular*[simp]: *regular* $\emptyset$
 ⟨proof⟩

lemma *empty-trans*[simp]: *transM* $\emptyset$
 ⟨proof⟩

lemma *emp-natM*[simp]: *natM* $\emptyset$
 ⟨proof⟩

lemma *binunion-emp*[simp]: $\emptyset \cup_M t = t \cup_M \emptyset = t$
 ⟨proof⟩

end

1.3.4 Set pair

Pairing is a simple set construction which can be easily obtained by set successor and empty set. It is interesting on its own, in particular in a context without the empty set axiom.

locale *L-pair* = *set-signature* +
assumes *pair*: $\forall x y. \exists z. \forall v. v \in z \longleftrightarrow v = x \vee v = y$

locale *L-setext-pair* = *L-setext* + *L-pair*

begin

lemma *singleton-def*'[*set-defs*]: $u \in \{y\}_M \longleftrightarrow u = y$
 ⟨proof⟩

lemma *singleton-eq*[*set-defs*]: $u = \{y\}_M \longleftrightarrow (\forall v. v \in u \longleftrightarrow v = y)$
 ⟨proof⟩

lemma *pair-def*'[*set-defs*]: $u \in \{x, y\}_M \longleftrightarrow u = x \vee u = y$
 ⟨proof⟩

lemma *pair-eq* [*set-defs*]: $w = \{x, y\}_M \longleftrightarrow (\forall u. (u \in w) \longleftrightarrow u = x \vee u = y)$

$\langle proof \rangle$

lemma *regular-not-self-mem*: **assumes** *regular* *x* **shows** $\neg x \in x$
 $\langle proof \rangle$

lemma *ordered-pair-unique[simp]*: $\langle u, v \rangle = \langle u', v' \rangle \longleftrightarrow u = u' \wedge v = v'$
 $\langle proof \rangle$

lemma *sing-eq-iff*: $\{a\}_M = \{b\}_M \longleftrightarrow a = b$
 $\langle proof \rangle$

lemma *sing-eq-pair-iff*: $\{a\}_M = \{b, c\}_M \longleftrightarrow a = b \wedge b = c$
 $\langle proof \rangle$

lemma *sing-eq-pair-iff'*: $\{b, c\}_M = \{a\}_M \longleftrightarrow a = b \wedge b = c$
 $\langle proof \rangle$

lemma *SFP-mem-ordered-pair[SFP-rules]*: *SetFormulaPredicate* $(\lambda \Xi. \Xi k \in \langle \Xi l, \Xi m \rangle)$
 $\langle proof \rangle$

lemma *SFP-eq-ordered-pair [SFP-rules]*: *SetFormulaPredicate* $(\lambda \Xi. \Xi k = \langle \Xi l, \Xi m \rangle)$
 $\langle proof \rangle$

lemma *SFP-ordered-pair-mem[SFP-rules]*: *SetFormulaPredicate* $(\lambda \Xi. \langle \Xi k, \Xi l \rangle \in \Xi m)$
 $\langle proof \rangle$

end

1.3.5 Set successor

Also known as adjunction. Enables constructing sets in steps. This axiom plays an important role in the fragment of set theory axiomatized by extensionality, empty set and set successor (see *L-setext-empty-setsuc* below), mutually interpretable with Robinson's arithmetic Q.

locale *L-setsuc* = *set-signature* +
assumes *setsuc*: $\forall x y. \exists z. \forall u. u \in z \longleftrightarrow u \in x \vee u = y$

locale *L-setext-setsuc* = *L-setext* + *L-setsuc*
 — some statements do not need the empty set

begin

lemma *setsuc-def'[set-defs, simp]*: $u \in \text{setsuc } M \ x \ y \longleftrightarrow (u \in x \vee u = y)$
 $\langle proof \rangle$

lemma *setsuc-triv*[*simp*]: $y \varepsilon x \implies \text{setsucM } x \ y = x$
 $\langle \text{proof} \rangle$

lemma *is-suc-def'*: $\text{is-sucM } x \longleftrightarrow (\exists \ y. \ x = \text{setsucM } y \ y)$
 $\langle \text{proof} \rangle$

lemma *trans-suc-trans*: $\text{transM } x \implies \text{transM } (\text{setsucM } x \ x)$
 $\langle \text{proof} \rangle$

lemma *epschain-suc-epschain*: $\text{epschain } x \implies \text{epschain } (\text{setsucM } x \ x)$
 $\langle \text{proof} \rangle$

lemma *ord-pred-fun*: **assumes** $\text{ordM } x \ \text{ordM } y$ $\text{setsucM } x \ x = \text{setsucM } y \ y$
shows $x = y$
 $\langle \text{proof} \rangle$

lemma *SFP-setsuc-mem*[*SFP-rules*]: *SetFormulaPredicate* $(\lambda \Xi. \ \Xi \ k \ \varepsilon \ \text{setsucM } (\Xi \ l) \ (\Xi \ m))$
 $\langle \text{proof} \rangle$

lemma *SFP-setsuc-eq*[*SFP-rules*]: *SetFormulaPredicate* $(\lambda \Xi. \ \Xi \ k = \text{setsucM } (\Xi \ l) \ (\Xi \ m))$
 $\langle \text{proof} \rangle$

end

locale *L-empty-setsuc* = *L-empty* + *L-setsuc*
— some statements do not need extensionality

begin

sublocale *L-pair*
 $\langle \text{proof} \rangle$

lemma *singleton-setsuc*: **shows** $\exists \ y. \ \forall \ u. \ u \ \varepsilon \ y \longleftrightarrow u = z$
 $\langle \text{proof} \rangle$

lemma *pair-setsuc*: **shows** $\exists \ y. \ \forall \ u. \ u \ \varepsilon \ y \longleftrightarrow u = z_1 \vee u = z_2$
 $\langle \text{proof} \rangle$

lemma *triple-setsuc*: **shows** $\exists \ y. \ \forall \ u. \ u \ \varepsilon \ y \longleftrightarrow u = z_1 \vee u = z_2 \vee u = z_3$
 $\langle \text{proof} \rangle$

lemma *regular-antisym-mem*: **assumes** *regular* $x \ z_1 \ \varepsilon \ x \ z_2 \ \varepsilon \ x$ **shows** $\neg (z_1 \ \varepsilon \ z_2 \wedge z_2 \ \varepsilon \ z_1)$
 $\langle \text{proof} \rangle$

lemma *regular-antisym2-mem*: **assumes** *regular* $x \ z_1 \ \varepsilon \ x \ z_2 \ \varepsilon \ x \ z_3 \ \varepsilon \ x$ **shows** $\neg$

$(z_1 \varepsilon z_2 \wedge z_2 \varepsilon z_3 \wedge z_3 \varepsilon z_1)$
 $\langle proof \rangle$

lemma *ord-mem-ord*: **assumes** $ordM\ v\ u \varepsilon v$ **shows** $ordM\ u$
 $\langle proof \rangle$

end

locale *L-setext-empty-setsuc* = *L-setext* + *L-empty* + *L-setsuc*

begin

sublocale *L-setext-empty*
 $\langle proof \rangle$

sublocale *L-setext-setsuc*
 $\langle proof \rangle$

sublocale *L-empty-setsuc*
 $\langle proof \rangle$

sublocale *L-setext-pair*
 $\langle proof \rangle$

lemma *empty-mem-ord*: **assumes** $ordM\ x\ x \neq \emptyset$ **shows** $\emptyset \varepsilon x$
 $\langle proof \rangle$

lemma *SP-injective* [*simp*]: *SetProperty* $(\lambda x. \forall a\ b\ c. \langle b, a \rangle \varepsilon x \wedge \langle c, a \rangle \varepsilon x \longrightarrow b = c)$
 $\langle proof \rangle$

lemma *SP-unique-image* [*simp*]: *SetProperty* $(\lambda x. \forall a\ b\ c. \langle a, b \rangle \varepsilon x \wedge \langle a, c \rangle \varepsilon x \longrightarrow b = c)$
 $\langle proof \rangle$

lemma *SFP-compose* [*simp, intro*]: *SetFormulaPredicate* $(\lambda \Xi. \exists a\ b\ c. \langle a, b \rangle \varepsilon \Xi\ k \wedge \langle b, c \rangle \varepsilon \Xi\ l \wedge \Xi\ m = \langle a, c \rangle)$
 $\langle proof \rangle$

lemma *SP-function* [*simp*]: *SetProperty* $(\lambda x. functionM\ x)$
 $\langle proof \rangle$

lemma *SP-one-one* [*simp*]: *SetProperty* $(\lambda x. one-oneM\ x)$
 $\langle proof \rangle$

lemma *dom-SR* [*simp*]: *SetRelation* $(\lambda x\ u. \exists w. x = \langle u, w \rangle)$
 $\langle proof \rangle$

lemma *rng-SR* [*simp*]: *SetRelation* $(\lambda x\ u. \exists w. x = \langle w, u \rangle)$

$\langle proof \rangle$

end

locale $L\text{-binunion} = \text{set-signature} +$
assumes $\text{binunion}: \forall x y. \exists z. \forall v. v \varepsilon z \longleftrightarrow (v \varepsilon x \vee v \varepsilon y)$

locale $L\text{-setext-binunion} = L\text{-setext} + L\text{-binunion}$

begin

lemma $\text{binunion-def}[\text{set-defs}]: u \varepsilon x \cup_M y \longleftrightarrow u \varepsilon x \vee u \varepsilon y$
 $\langle proof \rangle$

end

locale $L\text{-setext-pair-binunion} = L\text{-setext} + L\text{-pair} + L\text{-binunion}$
 — A specific minimalist combination of axioms that guarantees the existence of both ordered pair and set successor

begin

sublocale $L\text{-setext-binunion}$
 $\langle proof \rangle$

sublocale $L\text{-setext-pair}$
 $\langle proof \rangle$

sublocale $L\text{-setsuc}$
 $\langle proof \rangle$

sublocale $L\text{-setext-setsuc}$
 $\langle proof \rangle$

lemma $\text{SFP-ordered-pair-setsuc-eq}[\text{SFP-rules}]: \text{SetFormulaPredicate } (\lambda \Xi. \Xi k = \langle \Xi l, \text{setsucM } (\Xi m) (\Xi n) \rangle)$
 $\langle proof \rangle$

end

1.3.6 Regularity

locale $L\text{-reg} = \text{set-signature} +$
assumes $\text{reg}: \forall x. (\exists y. y \varepsilon x) \longrightarrow (\exists y. y \varepsilon x \wedge (\forall v. \neg (v \varepsilon y \wedge v \varepsilon x)))$

begin

lemma $\text{any-reg}[\text{simp}]: \text{regular } x$

<proof>

Note that $\neg x \varepsilon x$ cannot be proved without existence of a singleton.

end

lemma (in *set-signature*) *reg-all-regular-iff*:

$L\text{-reg } (\varepsilon) \longleftrightarrow (\forall x. \text{regular } x)$

<proof>

Schema of regularity

locale *L-regs* = *set-signature* +

assumes *regs*: *SetFormulaPredicate* $P \implies (\exists x. P (\exists (0:=x))) \longrightarrow (\exists x. P (\exists (0:=x)) \wedge (\forall y. y \varepsilon x \longrightarrow \neg P (\exists (0:=y))))$

begin

regs is stronger than normal regularity in the absence of infinity axiom, since one cannot prove the existence of transitive supersets/closures. See *not-reg-setind-implies-regs* in *ZFfin-regs-model.thy*

sublocale *L-reg*

<proof>

lemma *regs-epsind*:

SetFormulaPredicate $Q \implies (\forall x. (\forall y. (y \varepsilon x \longrightarrow Q (\exists (0:=y)))) \longrightarrow Q (\exists (0:=x))) \longrightarrow (\forall x. Q (\exists (0:=x)))$

<proof>

lemma *regs-mem-not-refl*: $\neg u \varepsilon u$

<proof>

end

Epsilon induction schema is logically equivalent to the regularity schema (that is, regardless of what ε means), see $((\exists x. ?P x) \longrightarrow (\exists x. ?B x \wedge ?P x)) = ((\forall x. ?B x \longrightarrow \neg ?P x) \longrightarrow (\forall x. \neg ?P x))$ above.

locale *L-epsind* = *set-signature* +

assumes *epsind*: *SetFormulaPredicate* $Q \implies (\forall x. (\forall y. (y \varepsilon x \longrightarrow Q (\exists (0:=y)))) \longrightarrow Q (\exists (0:=x))) \longrightarrow (\forall x. Q (\exists (0:=x)))$

begin

lemma *epsind-regs*: **assumes** *SetFormulaPredicate* $P \exists x. P (\exists (0:=x))$

shows $\exists x. P (\exists (0:=x)) \wedge (\forall y. y \varepsilon x \longrightarrow \neg P (\exists (0:=y)))$

<proof>

sublocale *L-regs*

<proof>

end

1.3.7 Separation

Separation is often an alternative to set successor. It allows one to construct “from above” many sets that set successor builds “from below”.

locale *L-sep* = *set-signature* +

assumes *sep*: *SetFormulaPredicate* $P \implies \forall x. \exists z. \forall u. (u \in z \longleftrightarrow u \in x \wedge P (\exists (\theta := u)))$

begin

sublocale *L-empty*

<proof>

lemma *sep-var*: **assumes** *SetFormulaPredicate* P **shows** $\forall x. \exists z. \forall u. (u \in z \longleftrightarrow u \in x \wedge P (\exists (n := u)))$

<proof>

lemma *sep-SP*: **assumes** *SetProperty* P **shows** $\forall x. \exists z. \forall u. (u \in z \longleftrightarrow u \in x \wedge P u)$

<proof>

lemma *sep-SR*: **assumes** *SetRelation* P **shows** $\forall x. \exists z. \forall u. (u \in z \longleftrightarrow u \in x \wedge P u r)$

<proof>

lemma *singleton-sep*: **assumes** $z \in x$

shows $\exists y. \forall u. u \in y \longleftrightarrow u = z$

<proof>

lemma *pair-sep*: **assumes** $z_1 \in x \ z_2 \in x$

shows $\exists y. \forall u. u \in y \longleftrightarrow u = z_1 \vee u = z_2$

<proof>

lemma *triple-sep*: **assumes** $z_1 \in x \ z_2 \in x \ z_3 \in x$

shows $\exists y. \forall u. u \in y \longleftrightarrow u = z_1 \vee u = z_2 \vee u = z_3$

<proof>

lemma *regular-not-self-mem-sep*: **assumes** *regular* x **shows** $\neg x \in x$

<proof>

lemma *regular-antisym-mem-sep*: **assumes** *regular* $x \ y_1 \in x \ y_2 \in x$ **shows** $\neg (y_1 \in y_2 \wedge y_2 \in y_1)$

<proof>

lemma *regular-antisym2-mem-sep*: **assumes** *regular* $x \ y_1 \in x \ y_2 \in x \ y_3 \in x$ **shows** $\neg (y_1 \in y_2 \wedge y_2 \in y_3 \wedge y_3 \in y_1)$

<proof>

lemma *ord-mem-ord-sep*: **assumes** *ordM* $v \ u \in v$ **shows** *ordM* u

$\langle proof \rangle$

end

The interest of this locale is to investigate intersections of various kinds.

locale $L\text{-setext-sep} = L\text{-setext} + L\text{-sep}$

begin

sublocale $L\text{-setext-empty}$

$\langle proof \rangle$

lemma separ-def-SFP : **assumes** $\text{SetFormulaPredicate } P$

shows $u \varepsilon \text{separationM } y (\lambda x. P(\Xi(n:=x))) \longleftrightarrow (u \varepsilon y \wedge (P(\Xi(n:=u))))$

$\langle proof \rangle$

lemma separ-def-SP : **assumes** $\text{SetProperty } P$

shows $u \varepsilon \text{separationM } y (\lambda x. P x) \longleftrightarrow (u \varepsilon y \wedge P u)$

$\langle proof \rangle$

lemma separ-def-SR : **assumes** $\text{SetRelation } P$

shows $u \varepsilon \text{separationM } y (\lambda x. P x r) \longleftrightarrow (u \varepsilon y \wedge P u r)$

$\langle proof \rangle$

lemma least-def' :

assumes $Q(\Xi(n:=w))$ **and** $\text{SetFormulaPredicate } Q$

shows $u \varepsilon \bigcap_M (\lambda x. Q(\Xi(n:=x))) \longleftrightarrow (\forall y. Q(\Xi(n:=y)) \longrightarrow u \varepsilon y)$

$\langle proof \rangle$

lemma least-def-ex :

$\exists w. Q(\Xi(n:=w)) \implies \text{SetFormulaPredicate } Q \implies$

$u \varepsilon \bigcap_M (\lambda x. Q(\Xi(n:=x))) \longleftrightarrow (\forall y. Q(\Xi(n:=y)) \longrightarrow u \varepsilon y)$

$\langle proof \rangle$

lemma $\text{intersection-def'[set-defs]}$: $z \varepsilon x \cap_M y \longleftrightarrow z \varepsilon x \wedge z \varepsilon y$

$\langle proof \rangle$

lemma intersect-subset : $x \cap_M y \subseteq_M x \cap_M y \subseteq_M y$

$\langle proof \rangle$

lemma intersect-regular : **assumes** $\text{regular } x \text{ regular } y$

shows $\text{regular } (x \cap_M y)$

$\langle proof \rangle$

lemma intersect-transM : **assumes** $\text{transM } x \text{ transM } y$

shows $\text{transM } (x \cap_M y)$

$\langle proof \rangle$

lemma $\text{intersect-epschain}$: **assumes** $\text{epschain } x \text{ epschain } y$

shows *epschain* $(x \cap_M y)$
 $\langle proof \rangle$

lemma *intersect-ordM*: **assumes** *ordM* x *ordM* y
shows *ordM* $(x \cap_M y)$
 $\langle proof \rangle$

lemma *ordM-subset-mem*: **assumes** *ordM* x *ordM* y $y \subset_M x$
shows $y \in x$
 $\langle proof \rangle$

lemma *ordM-total*: **assumes** *ordM* x *ordM* y
shows $x \in y \vee y \in x \vee x = y$
 $\langle proof \rangle$

lemma *nat-mem-nat*: **assumes** *natM* v $u \in v$ **shows** *natM* u
 $\langle proof \rangle$

lemma *set-of-ords-regular*: **assumes** $\forall y. y \in s \longrightarrow \text{ordM } y$
shows *regular* s
 $\langle proof \rangle$

lemma *intersect-natM*: **assumes** *natM* x *natM* y
shows *natM* $(x \cap_M y)$
 $\langle proof \rangle$

end

locale *L-setext-setsuc-sep* = *L-setext* + *L-setsuc* + *L-sep*

begin

We want to prove that ordinals are closed under the successor operation. However, regularity is a problem. It is not necessarily preserved by taking successors.

Separation is needed to avoid the following pathological situation: consider an epschain $C = \{c_z \mid z \in \mathbb{Z}\}$ of type $\mathbb{Z}$ where moreover $\emptyset \in c_z$ for each z . That is, $u \in c_m$ iff $u = c_k$ with $k < m$, or $u = \emptyset$. We postulate that an infinite subclass of C is a set iff it contains c_0 . Now, c_z are ordinals for all $z \leq 0$. But c_1 is not regular, since it contains an infinite descending chain $\{c_k \mid k \leq 0\}$ as a subset.

sublocale *L-setext-empty-setsuc*
 $\langle proof \rangle$

sublocale *L-setext-sep*
 $\langle proof \rangle$

lemma *regular-setsuc*: **assumes** *transM* x *regular* x *regular* y

shows *regular* (*setsucM* *x* *y*)
 ⟨*proof*⟩

lemma *ord-suc-ord*: **assumes** *ordM* *x* **shows** *ordM* (*setsucM* *x* *x*)
 ⟨*proof*⟩

lemma *nat-suc-nat*: **assumes** *natM* *x* **shows** *natM* (*setsucM* *x* *x*)
 ⟨*proof*⟩

lemma *one-natM[simp]*: *natM* ($\{\emptyset\}_M$)
 ⟨*proof*⟩

lemma *ord-mem-suc*: **assumes** *ordM* *x* **and** $y \in x$
shows *setsucM* *y* $y \in x \vee \text{setsucM } y \ y = x$
 ⟨*proof*⟩

lemma *ord-limit-or-suc*: **assumes** *ordM* *x*
shows *is-sucM* *x* $\vee (\forall \ u. \ u \in x \longrightarrow \text{setsucM } u \ u \in x)$
 ⟨*proof*⟩

lemma **assumes** *regular* *x* *regular* *y*
shows *regular* (*setsucM* *x* *y*)
 ⟨*proof*⟩

end

1.3.8 Transitive superset

The existence of transitive supersets needs to be assumed explicitly if the axiom of infinity is absent (or negated). It is related to regularity, since it enables obtaining an infinite set from infinite descending epsilon chains.

locale *L-ts* = *set-signature* +
assumes *ts*: $\forall \ x. \ \exists \ z. \ (\text{transM } z \wedge (x \subseteq_M z))$

locale *L-setext-sep-ts* = *L-setext* + *L-sep* + *L-ts*

begin

sublocale *L-setext-sep*
 ⟨*proof*⟩

lemma *least-ts-def'*: $u \in \text{least-tsM } x \longleftrightarrow (\forall \ v. \ \text{transM } v \wedge x \subseteq_M v \longrightarrow u \in v)$
 (**is** $u \in \text{least-tsM } x \longleftrightarrow (\forall \ v. \ ?Q \ v \longrightarrow u \in v)$)
 ⟨*proof*⟩

lemma *least-ts-is-transitive*: *transM* (*least-tsM* *x*)
 ⟨*proof*⟩

end

locale $L\text{-setext-sep-reg} = L\text{-setext} + L\text{-sep} + L\text{-reg}$

locale $L\text{-setext-sep-reg-ts} = L\text{-setext} + L\text{-sep} + L\text{-reg} + L\text{-ts}$

begin

sublocale $L\text{-setext-sep-ts}$
 $\langle \text{proof} \rangle$

The schema of regularity follows from regularity and transitive superset.

sublocale $L\text{-regsch}$
 $\langle \text{proof} \rangle$

end

1.3.9 Replacement

locale $L\text{-repl} = \text{set-signature} +$

assumes $\text{replf: } \text{SetFormulaPredicate } P \implies (\forall u. \exists! v. P (\Xi(\theta:=u, l:=v))) \longrightarrow$
 $(\forall x. \exists z. \forall v. v \varepsilon z \longleftrightarrow (\exists u. u \varepsilon x \wedge P (\Xi(\theta:=u, l:=v))))$

— We can replace x_0 by the unique x_1 satisfying $P(x_0, x_1)$. Note the presence of parameters $x_i, i \geq 2$

locale $L\text{-setext-empty-repl} = L\text{-setext} + L\text{-empty} + L\text{-repl}$

begin

Replacing using total functions is equivalent to replacing using partial functions.

lemma $\text{replp: assumes } \text{SetFormulaPredicate } P \text{ and func: } \forall u v w. P (\Xi(\theta:=u, l:=v)) \wedge P (\Xi(\theta:=u, l:=w)) \longrightarrow v = w$
shows $\forall x. \exists z. (\forall v. v \varepsilon z \longleftrightarrow (\exists u. u \varepsilon x \wedge P (\Xi(\theta:=u, l:=v))))$
 $\langle \text{proof} \rangle$

lemma $\text{replp-vars: assumes } \text{SetFormulaPredicate } P \text{ and func: } \forall u v w. P (\Xi(k:=u, l:=v)) \wedge P (\Xi(k:=u, l:=w)) \longrightarrow v = w$
shows $\forall x. \exists z. (\forall v. v \varepsilon z \longleftrightarrow (\exists u. u \varepsilon x \wedge P (\Xi(k:=u, l:=v))))$
 $\langle \text{proof} \rangle$

lemma $\text{repl-SR: assumes } \text{SetRelation } P \forall u v w. P u v \wedge P u w \longrightarrow v = w$
shows $\forall x. \exists z. (\forall v. v \varepsilon z \longleftrightarrow (\exists u. u \varepsilon x \wedge P u v))$
 $\langle \text{proof} \rangle$

lemma replace-def' :

assumes $\text{SetFormulaPredicate } P \forall u v w. P (\Xi(k:=u, l:=v)) \wedge P (\Xi(k:=u, l:=w)) \longrightarrow v = w$
shows $\forall v. v \varepsilon \text{replaceM } (\lambda u v. P (\Xi(k:=u, l:=v))) x \longleftrightarrow (\exists u. u \varepsilon x \wedge (P (\Xi(k:=u, l:=v))))$

<proof>

Separation can be obtained from replacement.

sublocale *L-setext-sep*

<proof>

end

Separation does not follow from replacement without the empty set axiom.
We show it by constructing a counter-model.

theorem *not-repl-ext-implies-separ*: $\neg (\forall \text{ mem. } L\text{-repl mem} \wedge L\text{-setext mem} \longrightarrow L\text{-sep mem})$

<proof>

locale *L-setext-empty-setsuc-repl* = *L-setext* + *L-empty* + *L-setsuc* + *L-repl*

begin

sublocale *L-setext-empty-setsuc*

<proof>

sublocale *L-setext-empty-repl*

<proof>

lemma *tarski-setsuc-tarski*: **assumes** *tarski-fin x* **shows** *tarski-fin (setsucM x y)*

<proof>

end

1.3.10 Powerset

locale *L-power* = *set-signature* +

assumes

power: $\forall x. \exists y. (\forall u. u \varepsilon y \longleftrightarrow u \subseteq_M x)$

locale *L-setext-empty-power* = *L-setext* + *L-empty* + *L-power*

begin

sublocale *L-setext-empty*

<proof>

lemma *powerset-def'[set-defs]*: $u \varepsilon (\mathfrak{P} x) \longleftrightarrow u \subseteq_M x$

<proof>

lemma $\emptyset \varepsilon \mathfrak{P} \emptyset$

<proof>

lemma *set-one-mem* [*simp*]: $u \in \mathfrak{P} \emptyset \longleftrightarrow u = \emptyset$
 $\langle proof \rangle$

lemma *set-one-def*: $\mathfrak{P} \emptyset = \{\emptyset\}_M$
 $\langle proof \rangle$

lemma *set-two-mem*: $u \in \mathfrak{P} (\mathfrak{P} \emptyset) \longleftrightarrow u = \emptyset \vee u = \{\emptyset\}_M$
 $\langle proof \rangle$

end

locale *L-setext-empty-power-repl* = *L-setext* + *L-empty* + *L-power* + *L-repl*

— Subsumes many already existing locales. In particular, we obtain set successor from powerset and replacement

begin

sublocale *L-setext-empty-repl*
 $\langle proof \rangle$

sublocale *L-setext-empty-power*
 $\langle proof \rangle$

sublocale *L-setext-empty-setsuc*
 $\langle proof \rangle$

sublocale *L-setext-empty-setsuc-repl*
 $\langle proof \rangle$

end

1.3.11 Union

locale *L-union* = *set-signature* +
assumes *union*: $\forall x. \exists y. \forall v. v \in y \longleftrightarrow (\exists u. u \in x \wedge v \in u)$

locale *L-setext-union* = *L-setext* + *L-union*

begin

lemma *union-def*'[*set-defs*]: $u \in \bigcup_M x \longleftrightarrow (\exists v. v \in x \wedge u \in v)$
 $\langle proof \rangle$

lemma *union-trans-trans*: **assumes** $\forall v. v \in x \longrightarrow \text{transM } v$
shows $\text{transM } (\bigcup_M x)$
 $\langle proof \rangle$

end

locale $L\text{-setext-sep-binunion-ts} = L\text{-setext} + L\text{-sep} + L\text{-binunion} + L\text{-ts}$

begin

sublocale $L\text{-setext-binunion}$
 $\langle \text{proof} \rangle$

sublocale $L\text{-setext-sep-ts}$
 $\langle \text{proof} \rangle$

lemma trans-union : $\text{trans}M\ u \implies \text{trans}M\ v \implies \text{trans}M\ (u \cup_M v)$
 $\langle \text{proof} \rangle$

lemma leastTS-union : $\text{least-ts}M\ (u \cup_M v) = \text{least-ts}M\ u \cup_M \text{least-ts}M\ v$
 $\langle \text{proof} \rangle$

end

The following locale is called Elementary Set Theory (EST) in Baratella and Ferro, "A theory of Sets with the Negation of the Axiom of Infinity", 1993.

locale $L\text{-setext-empty-union-repl-pair} = L\text{-setext} + L\text{-empty} + L\text{-union} + L\text{-repl} + L\text{-pair}$

begin

— binunion is the union of a pair

sublocale $L\text{-setext-pair-binunion}$
 $\langle \text{proof} \rangle$

end

The locale $L\text{-setext-empty-union-repl-pair}$ (i.e., the fragment EST also used by Baratella and Ferro [1]) is the right context in which to show that the schema of regularity yields existence of transitive closures (least transitive supersets)

locale $L\text{-setext-empty-union-repl-pair-regs} = L\text{-setext} + L\text{-empty} + L\text{-union} + L\text{-repl} + L\text{-pair} + L\text{-regs}$

begin

sublocale $L\text{-setext-empty-union-repl-pair}$
 $\langle \text{proof} \rangle$

sublocale $L\text{-setext-empty-repl}$
 $\langle \text{proof} \rangle$

sublocale $L\text{-setext-union}$
 $\langle \text{proof} \rangle$

lemma *transitive-closure-ex*:

shows $\forall x. \exists z. x \subseteq_M z \wedge \text{trans}M\ z \wedge (\forall u. u \varepsilon z \longleftrightarrow (\forall t. x \subseteq_M t \wedge \text{trans}M\ t \longrightarrow u \varepsilon t))$
(is $\forall x. \exists z. ?TC\ x\ z$ **is** $\forall x. ?hasTC\ x$)
 $\langle proof \rangle$

sublocale *L-setext-sep-reg-ts*
 $\langle proof \rangle$

end

locale *L-setext-empty-power-union-repl* = *L-setext* + *L-empty* + *L-power* + *L-union* + *L-repl*

— ZF without infinity and regularity. Axioms enabling reasoning about functions, cardinality, and therefore about Dedekind finiteness.

begin

sublocale *L-setext-empty-power-repl*
 $\langle proof \rangle$

sublocale *L-setext-setsuc-sep*
 $\langle proof \rangle$

sublocale *L-setext-union*
 $\langle proof \rangle$

sublocale *L-setext-empty-union-repl-pair*
 $\langle proof \rangle$

lemma *ordered-pair-mem*: **assumes** $v \varepsilon x\ v' \varepsilon y$ **shows** $\langle v, v' \rangle \varepsilon \mathfrak{P}(\mathfrak{P}(x \cup_M y))$
 $\langle proof \rangle$

lemma *ordered-pair-mem-union*: **assumes** $\langle u, v \rangle \varepsilon r$ **shows** $u \varepsilon \bigcup_M (\bigcup_M r)\ v \varepsilon \bigcup_M (\bigcup_M r)$
 $\langle proof \rangle$

lemma *car-prod-ex*: $\exists z. \forall u. u \varepsilon z \longleftrightarrow (\exists v\ v'. v \varepsilon x \wedge v' \varepsilon y \wedge u = \langle v, v' \rangle)$
 $\langle proof \rangle$

lemma *car-prod-def'*: $u \varepsilon x \times_M y \longleftrightarrow (\exists v\ v'. v \varepsilon x \wedge v' \varepsilon y \wedge u = \langle v, v' \rangle)$
 $\langle proof \rangle$

lemma *rel-inv-def'*: $u \varepsilon \text{rel-inverse}M\ r \longleftrightarrow (\exists a\ b. \langle a, b \rangle \varepsilon r \wedge u = \langle b, a \rangle)$
 $\langle proof \rangle$

lemma *dom-def'*: $u \varepsilon \text{dom}M\ r \longleftrightarrow (\exists v. \langle u, v \rangle \varepsilon r)$

$\langle proof \rangle$

lemma *rng-def'*: $u \in \text{rng}M\ r \longleftrightarrow (\exists v. \langle v, u \rangle \in r)$
 $\langle proof \rangle$

lemma *SFP-dom[simp, intro]*: *SetFormulaPredicate* $(\lambda \Xi. \Xi\ k = \text{dom}M\ (\Xi\ l))$
 $\langle proof \rangle$

lemma *SFP-rng[simp, intro]*: *SetFormulaPredicate* $(\lambda \Xi. \Xi\ k = \text{rng}M\ (\Xi\ l))$
 $\langle proof \rangle$

lemmas[*SFP-rules*] = *SFP-rng*[*unfolded set-defs logsimps*] *SFP-dom*[*unfolded set-defs logsimps*]

lemma *SetRelation* $(\lambda x\ y. \exists f. \text{one-one}M\ f \wedge x = \text{dom}M\ f \wedge y = \text{rng}M\ f)$
 $(\text{is } \text{SetRelation } (\lambda x\ y. (\exists f. ?P\ x\ y\ f)))$
 $\langle proof \rangle$

lemma *SR-equiv[simp]*: *SetRelation* $(\lambda x\ y. x \approx_M y)$
 $\langle proof \rangle$

lemma *SP-dedekind[simp]*: *SetProperty dedekind-fin*
 $\langle proof \rangle$

lemma *SR-card*: *SetRelation cardinality*
 $\langle proof \rangle$

lemma *rel-inv-dom-rng*: $\text{dom}M\ (\text{rel-inverse}M\ r) = \text{rng}M\ r\ \text{rng}M\ (\text{rel-inverse}M\ r)$
 $= \text{dom}M\ r$
 $\langle proof \rangle$

lemma *rel-inv-rel*: $\text{relation}M\ f \implies \text{relation}M\ (\text{rel-inverse}M\ f)$
 $\langle proof \rangle$

lemma *one-one-inv*: $\text{one-one}M\ f \implies \text{one-one}M\ (\text{rel-inverse}M\ f)$
 $\langle proof \rangle$

lemma *dedekind-setsuc-dedekind*: **assumes** *dedekind-fin x* **shows** *dedekind-fin (setsucM x t)*
 $\langle proof \rangle$

lemma *set-equivalent-sym*: $x \approx_M y \longleftrightarrow y \approx_M x$
 $\langle proof \rangle$

lemma *compose-def'*: $u \in f \circ_M g \longleftrightarrow (\exists a\ b\ c. \langle a, b \rangle \in g \wedge \langle b, c \rangle \in f \wedge u = \langle a, c \rangle)$
 $\langle proof \rangle$

lemma *rel-comp*: **assumes** *relationM f relationM g* **shows** *relationM (f $\circ_M$ g)*
 $\langle proof \rangle$

lemma *fun-comp*: **assumes** *functionM* *f* *functionM* *g* **shows** *functionM* (*f* $\circ_M$ *g*)
 $\langle proof \rangle$

lemma *one-one-comp*: **assumes** *one-oneM* *f* *one-oneM* *g* **shows** *one-oneM* (*f* $\circ_M$ *g*)
 $\langle proof \rangle$

lemma *set-equivalent-trans*: **assumes** $x \approx_M y$ $y \approx_M z$ **shows** $x \approx_M z$
 $\langle proof \rangle$

lemma *union-of-ords-regular*: **assumes** $\forall y. y \in s \longrightarrow \text{ordM } y$
shows *regular* ($\bigcup_M s$)
 $\langle proof \rangle$

lemma *union-of-ords-ord*: **assumes** $\forall y. y \in s \longrightarrow \text{ordM } y$
shows *ordM* ($\bigcup_M s$)
 $\langle proof \rangle$

lemma *union-ord*: **assumes** *ordM* *x* **shows** *ordM* ($\bigcup_M x$)
 $\langle proof \rangle$

lemma *non-limit-union-ord-mem*: **assumes** $\forall u. u \in s \longrightarrow \text{ordM } u$ *is-sucM* ($\bigcup_M s$)
shows $\bigcup_M s \in s$
 $\langle proof \rangle$

lemma *limit-ord*: **assumes** $\forall u. u \in s \longrightarrow \text{ordM } u \wedge \text{setsucM } u$ $u \in s$
shows $\neg \text{is-sucM } (\bigcup_M s)$
 $\langle proof \rangle$

lemma *not-limit-ord*: **assumes** *ordM* *x* *is-sucM* *x*
shows $x = \text{setsucM } (\bigcup_M x)$
 $\langle proof \rangle$

lemma *natM-fin*: **assumes** $s \subseteq_M x$ $s \neq \emptyset$ $\forall u. u \in s \longrightarrow \text{setsucM } u$ $u \in s$
shows $\neg \text{natM } x$
 $\langle proof \rangle$

lemma *nat-induction-sfp*: **assumes** *SetFormulaPredicate* *P* **and** $P (\exists (0:=\emptyset))$
and
step: $\bigwedge x. \text{natM } x \Longrightarrow P (\exists (0:=x)) \Longrightarrow P (\exists (0:=\text{setsucM } x))$ **and** *natM* *x*
shows $P (\exists (0:=x))$
 $\langle proof \rangle$

lemma *nat-induction-sp*: **assumes** *SetProperty* *P* **and** $P \emptyset$ **and** *natM* *x* **and**
step: $\bigwedge x. \text{natM } x \Longrightarrow P x \Longrightarrow P (\text{setsucM } x)$
shows $P x$
 $\langle proof \rangle$

theorem *nat-tarski-fin*: **assumes** *natM x* **shows** *tarski-fin x*
 ⟨*proof*⟩

lemma *nat-dedekind-finite*: **assumes** *natM x* **shows** *dedekind-fin x*
 ⟨*proof*⟩

lemma *card-emp*: *cardinality* $\emptyset$ $\emptyset$
 ⟨*proof*⟩

lemma *nat-equiv-unique*: **assumes** *natM x natM y* $x \approx_M y$
shows $x = y$
 ⟨*proof*⟩

lemma *card-fun*: **assumes** *cardinality x n cardinality x m* **shows** $n = m$
 ⟨*proof*⟩

end

locale *L-setext-empty-power-union-repl-reg* = *L-setext* + *L-empty* + *L-power* +
L-union + *L-repl*
 + *L-reg*

begin

Some consequences of the axiom of regularity

sublocale *L-setext-empty-power-union-repl*
 ⟨*proof*⟩

lemma *mem-not-refl*: $\neg x \varepsilon x$
 ⟨*proof*⟩

lemma *mem-antisym*: $\neg (u \varepsilon v \wedge v \varepsilon u)$
 ⟨*proof*⟩

lemma *suc-unique*: **assumes** *setsucM c c = setsucM d d*
shows $c = d$
 ⟨*proof*⟩

lemma *mem-neq-singleton*: $x \neq \{x\}_M$
 ⟨*proof*⟩

lemma *suc-subset[simp]*: $z \subset_M \text{setsucM } z$
 ⟨*proof*⟩

lemma *card-setsuc*: **assumes** $\neg y \varepsilon x$ *cardinality x m* **shows** *cardinality (setsucM*
x y) (setsucM m m)
 ⟨*proof*⟩

end

1.3.12 Successor induction

An important fragment of the theory of hereditarily finite sets. The finiteness principle used is the induction schema for set formulas. This route to axiomatizing ZF_{fin} is developed in Vopěnka: Mathematics in the alternative set theory [10]. Notice that no regularity principles are used in this fragment.

locale *L-setext-empty-setsuc-setind* = *L-setext* + *L-empty* + *L-setsuc* + *L-setind*

begin

sublocale *L-setext-empty-setsuc*

<proof>

lemma *binunion-ex*:

shows $\exists z. (\forall u. u \in z \longleftrightarrow u \in x \vee u \in x')$ (**is** *?P* *x x'*)
<proof>

sublocale *L-union*

<proof>

sublocale *L-repl*

<proof>

sublocale *L-setext-empty-repl*

<proof>

lemma *setsuc-separ*: **assumes** $y \in u$

shows $u = \text{setsucM } (\text{separationM } u (\lambda x. x \neq y)) y$
<proof>

lemma *setsuc-subset-setind*: $u \subseteq_M \text{setsucM } x y \longleftrightarrow u \subseteq_M x \vee (\exists u'. u' \subseteq_M x \wedge u = \text{setsucM } u' y)$

<proof>

sublocale *L-power*

<proof>

sublocale *L-setext-empty-power-union-repl*

<proof>

lemma *min-subset-ex*: **assumes** $u \neq \emptyset$ **shows** $\exists z. z \in u \wedge (\nexists w. w \in u \wedge w \subset_M z)$

<proof>

A general variant of Tarski finiteness axiom (also proved in Vopěnka's book). Nonempty sets have maximal (and minimal) elements under inclusion.

lemma *max-subset-ex*: **assumes** $u \neq \emptyset$ **shows** $\exists z. z \in u \wedge (\nexists w. w \in u \wedge z \subset_M w)$

w)
 $\langle proof \rangle$

lemma *max-mem*: **assumes** $P\ n\ n \in x\ SetProperty\ P$
shows $\exists\ m. m \in x \wedge P\ m \wedge (\forall\ k. k \in x \wedge P\ k \longrightarrow \neg m \subset_M k)$
 $\langle proof \rangle$

sublocale *L-tarski*
 $\langle proof \rangle$

sublocale *L-dedekind*
 $\langle proof \rangle$

lemma *fun-images*: **assumes** *SetFormulaPredicate* P **and** $\forall\ u. (\exists! v. P(\Xi(0:=u, 1:=v)))$
shows $\exists\ z. (\forall\ v. v \in z \longleftrightarrow (\exists\ u. u \in x \wedge P(\Xi(0:=u, 1:=v))))$
 $\langle proof \rangle$

sublocale *L-fin*
 $\langle proof \rangle$

theorem *ord-is-suc*: **assumes** $ordM\ x$ **shows** $x = \emptyset \vee is-sucM\ x$
 $\langle proof \rangle$

theorem *ord-iff-nat*: $ordM\ x \longleftrightarrow natM\ x$
 $\langle proof \rangle$

lemma *ord-iff-empty-or-suc*: $ordM\ x \longleftrightarrow x = \emptyset \vee (\exists\ y. ordM\ y \wedge x = setsucM\ y)$
 $\langle proof \rangle$

lemma *union-of-ords-mem*: **assumes** $\forall\ y. y \in s \longrightarrow ordM\ y\ s \neq \emptyset$
shows $\bigcup_M s \in s$
 $\langle proof \rangle$

end

1.3.13 Negation of inf

ZF with inf replaced by fin

locale *L-setext-empty-power-union-repl-reg-fin* = *L-setext* + *L-empty* + *L-power*
+ *L-union* + *L-repl* + *L-reg* + *L-fin*

begin

sublocale *L-setext-empty-power-union-repl-reg*
 $\langle proof \rangle$

sublocale *L-setind*
 $\langle proof \rangle$

lemma *cardinality-ex*: $\exists! n. \text{natM } n \wedge x \approx_M n$
 $\langle \text{proof} \rangle$

sublocale *L-setext-empty-setsuc-setind*
 $\langle \text{proof} \rangle$

end

1.3.14 Dedekind finite

locale *L-setext-empty-power-union-repl-dedekind* = *L-setext* + *L-empty* + *L-power* +
+ *L-union* + *L-repl* + *L-dedekind*

begin

sublocale *L-setext-empty-power-union-repl*
 $\langle \text{proof} \rangle$

sublocale *L-fin*
 $\langle \text{proof} \rangle$

end

1.3.15 Tarski finite

locale *L-setext-empty-power-sep-setsuc-tarski* = *L-setext* + *L-empty* + *L-power* +
+ *L-sep* + *L-setsuc* + *L-tarski*

begin

sublocale *L-setext-empty-power*
 $\langle \text{proof} \rangle$

sublocale *L-setext-empty-setsuc*
 $\langle \text{proof} \rangle$

In a suitable context, the axiom of Tarski finitness yields the set induction schema.

sublocale *L-setind*
 $\langle \text{proof} \rangle$

sublocale *L-setext-empty-setsuc-setind*
 $\langle \text{proof} \rangle$

It follows in particular that union and replacement are provable in this context.

sublocale *L-repl*
 $\langle \text{proof} \rangle$

sublocale *L-union*

<proof>

end

1.3.16 Set induction and regularity schema

An apparently minor variation of the set induction schema, which nevertheless yields also the schema of regularity (i.e., epsilon induction). It is used in Pudlák and Sochor [5].

locale *L-setindregs* = *set-signature* +

assumes *setindregs*: *SetFormulaPredicate* *P* $\implies$

$P(\Xi(0 := \emptyset)) \longrightarrow (\forall x y. P(\Xi(0 := x)) \wedge P(\Xi(0 := y)) \longrightarrow P(\Xi(0 := \text{setsuc } M x y))) \longrightarrow (\forall x. P(\Xi(0 := x)))$

begin

lemma *setindregs-sp*: **assumes** *SetProperty* *P* $P \emptyset \forall x y. P x \wedge P y \longrightarrow P(\text{setsuc } M x y)$

shows $\forall x. P x$

<proof>

lemma *setindregs-var*: **assumes** *SetFormulaPredicate* *P* $P(\Xi(n := \emptyset)) \forall x y. P(\Xi(n := x)) \wedge P(\Xi(n := y))$

$\longrightarrow P(\Xi(n := \text{setsuc } M x y))$

shows $\forall x. P(\Xi(n := x))$

<proof>

sublocale *L-setind*

<proof>

end

locale *L-setext-empty-setsuc-setindregs* = *L-setext* + *L-empty* + *L-setsuc* + *L-setindregs*

begin

sublocale *L-setext-empty-setsuc-setind*

<proof>

lemma *epsind-from-setindregs-sp*: **assumes** *spp*: *SetProperty* *P* **and** *indp*: $(\forall x. (\forall y. (y \in x \longrightarrow P y)) \longrightarrow P x)$

shows $\forall x. P x$

<proof>

lemma *epsind-from-setindregs*: **assumes** *sfp*: *SetFormulaPredicate* *P* **and** *indp*: $(\forall x. (\forall y. (y \in x \longrightarrow P(\Xi(0 := y)))) \longrightarrow P(\Xi(0 := x)))$

shows $\forall x. P(\Xi(0:=x))$
 $\langle proof \rangle$

sublocale $L\text{-epsind}$
 $\langle proof \rangle$

sublocale $L\text{-setext-empty-union-repl-pair-regschr}$
 $\langle proof \rangle$

end

1.3.17 Summary of dependencies

theorem $\text{epsind-regschr-iff}$:
 $L\text{-epsind mem} \longleftrightarrow L\text{-regschr mem}$
 $\langle proof \rangle$

We give additional names to some important collections of axioms. Here is a "canonical" axiomatization of the theory of hereditarily finite sets.

locale $ZFfn = L\text{-setext} + L\text{-empty} + L\text{-power} + L\text{-union} + L\text{-repl} + L\text{-fin} + L\text{-epsind}$

begin
sublocale $L\text{-setext-empty-power-union-repl-reg-fin}$
 $\langle proof \rangle$
sublocale $L\text{-regschr}$
 $\langle proof \rangle$
sublocale $L\text{-reg}$
 $\langle proof \rangle$
sublocale $L\text{-setsuc}$
 $\langle proof \rangle$
sublocale $L\text{-sep}$
 $\langle proof \rangle$
sublocale $L\text{-setind}$
 $\langle proof \rangle$
sublocale $L\text{-dedekind}$
 $\langle proof \rangle$
sublocale $L\text{-tarski}$
 $\langle proof \rangle$
end

This is the list of axioms for sets from Vopěnka's book [10].

locale $ASTset = L\text{-setext} + L\text{-empty} + L\text{-setsuc} + L\text{-setind} + L\text{-regschr}$

begin

Vopěnka [10] shows that all axioms of $ZFfn$ are provable in $ASTset$

sublocale $ZFfn$
 $\langle proof \rangle$

The variation of set induction which combines it with regularity schema is provable from set induction schema and epsilon induction (i.e., regularity schema). Taken for granted in [5].

sublocale $L\text{-setindregsch}$
 $\langle \text{proof} \rangle$

end

$ZFfin$ and $ASTset$ are two different axiomatizations of the same theory.

sublocale $ASTset \subseteq ZFfin$
 $\langle \text{proof} \rangle$
sublocale $ZFfin \subseteq ASTset$
 $\langle \text{proof} \rangle$

In the AST, set induction and epsilon induction (i.e., regularity schema) can be replaced by the variant *setindregsch*

theorem *setindregsch-ast*: $L\text{-setext-empty-setsuc-setindregsch mem} \longleftrightarrow ASTset mem$
 $\langle \text{proof} \rangle$

theorem *zffin-ast*: $ZFfin mem \longleftrightarrow ASTset mem$
 $\langle \text{proof} \rangle$

Ambivalence of the separation schema and the replacement schema.

theorem *repl-implies-sep*:
shows $L\text{-setext-empty-repl mem} \implies L\text{-sep mem}$
 $\langle \text{proof} \rangle$

Under certain finiteness assumptions, separation entails replacement. See [2] for details.

theorem *sep-implies-repl*:
shows $L\text{-setext-empty-power-sep-setsuc-tarski mem} \implies L\text{-repl mem}$
 $\langle \text{proof} \rangle$

Entailment between different finiteness principles, in their natural contexts.

The following is included in Vopěnka [10] by exploring the consequences of induction for set formulas.

theorem (in $L\text{-setext-empty-setsuc}$)
shows *setind-implies-tarski*: $L\text{-setind } (\varepsilon) \implies L\text{-tarski } (\varepsilon)$ **and**
setind-implies-fin-by-setsuc: $L\text{-setind } (\varepsilon) \implies L\text{-fin } (\varepsilon)$ **and**
setind-implies-dedekind-by-setsuc: $L\text{-setind } (\varepsilon) \implies L\text{-dedekind } (\varepsilon)$
 $\langle \text{proof} \rangle$

theorem (in $L\text{-setext-empty-power-union-repl}$)
dedekind-implies-fin: $L\text{-dedekind } (\varepsilon) \implies L\text{-fin } (\varepsilon)$
 $\langle \text{proof} \rangle$

Equivalence of finiteness principles, in sufficiently strong common context.

theorem (in *L-setext-empty-power-union-repl-reg*)
 $fin\text{-}implies\text{-}tarski: L\text{-}fin(\varepsilon) \implies L\text{-}tarski(\varepsilon)$ **and**
 $tarski\text{-}implies\text{-}fin: L\text{-}tarski(\varepsilon) \implies L\text{-}fin(\varepsilon)$ **and**
 $fin\text{-}implies\text{-}setind: L\text{-}fin(\varepsilon) \implies L\text{-}setind(\varepsilon)$ **and**
 $setind\text{-}implies\text{-}fin: L\text{-}setind(\varepsilon) \implies L\text{-}fin(\varepsilon)$ **and**
 $fin\text{-}implies\text{-}dedekind: L\text{-}fin(\varepsilon) \implies L\text{-}dedekind(\varepsilon)$
 $\langle proof \rangle$

The validity of the regularity schema/epsilon induction is closely related to the existence of a transitive superset. See also Proposition 5.4, Kaye and Wong [3]. There, the equivalence of *L-epsind* and *L-ts* is shown in the context of EST + regularity. Instead, we follow Sochor [7] and [8] in improving the claim by weakening both implications.

theorem (in *L-setext-sep-reg*) *ts-implies-epsind*:
 $L\text{-}ts(\varepsilon) \implies L\text{-}epsind(\varepsilon)$
 $\langle proof \rangle$

theorem (in *L-setext-empty-union-repl-pair*) *epsind-implies-ts*:
 $L\text{-}epsind(\varepsilon) \implies L\text{-}ts(\varepsilon)$
 $\langle proof \rangle$

Consequently, in a sufficiently strong context we can verify the equivalence of axioms A81 (regularity) + A82 (transitive superset) and A8 (schema of regularity) from Sochor [7], p. 709.

theorem (in *L-setext-empty-union-repl-pair*) *ts-reg-iff-regsch*:
 $L\text{-}ts(\varepsilon) \wedge L\text{-}reg(\varepsilon) \iff L\text{-}regsch(\varepsilon)$
 $\langle proof \rangle$

end

Chapter 2

Models and counter-models

```
theory ZFfin-HF
imports HereditarilyFinite.Rank ZFfin
begin
```

2.1 Hereditarily finite sets

We show that the hereditarily finite sets as implemented in *HereditarilyFinite.HF* are a model of *ZFfin* as implemented in *ZF-finite.ZFfin*

```
interpretation zfhf: ZFfin ( $\in$ )
  rewrites zfhf.emptysetM = 0 and
    zfhf.singletonM y =  $\{y\}$  and
    zfhf.setsucM x y =  $x \triangleleft y$ 
 $\langle$ proof $\rangle$ 

end
```

2.2 Permutation models

```
theory Permutation-models
imports ZFfin
begin
```

A useful tool of constructing models with specific properties is the Bernays-Rieger permutation method, which redefines the membership relation. We show that a permutation model obtained in this way preserves extensionality, empty set, powerset, union, and replacement. The method has been used, inter alia, in several works directly relevant to our formalization ([6], [11], [1], [4])

```
locale permutation-model = L-setext-empty-power-union-repl +
  fixes perm
  assumes
    bij-perm: bij (perm :: 'a  $\Rightarrow$  'a) and
```

SR-fmem: *SetRelation* ($\lambda x y. x \varepsilon \text{perm } y$)

begin

abbreviation *perm-mem* (**infixr** ε^f 51) **where**
perm-mem $x y \equiv x \varepsilon \text{perm } y$

sublocale *L-setext-empty*
 $\langle \text{proof} \rangle$

interpretation *pm*: *L-setext perm-mem*
 $\langle \text{proof} \rangle$

lemma *SFP-fmem[SFP-rules]*: *SetFormulaPredicate* ($\lambda \Xi. \Xi m \varepsilon^f \Xi n$)
 $\langle \text{proof} \rangle$

lemma *SR-perm-eq*: *SetRelation* ($\lambda x y. \text{perm } x = y$)
 $\langle \text{proof} \rangle$

lemma *permSFP-SFP*: **assumes** *pm.SetFormulaPredicate P* **shows** *SetFormulaPredicate P*
 $\langle \text{proof} \rangle$

lemma *perm-model*:
shows *L-setext-empty-power-union-repl perm-mem*
 $\langle \text{proof} \rangle$

end

end

2.3 Regularity

theory *Not-regular-model*
imports *Permutation-models*
begin

Axioms of extensionality, empty set, powerset, union, replacement and set induction are not sufficient to prove regularity, even in the presence of transitive superset axiom.

We prove this using permutation models. It is enough to transpose 0 and 1 in order to obtain a model which violates regularity, cf. [1, Theorem 13]

context *L-setext-empty-power-union-repl*

begin

definition *swap-zero-one* :: $'a \Rightarrow 'a$ **where**
swap-zero-one $x = (\text{if } x = \emptyset \text{ then } \{\emptyset\}_M \text{ else } (\text{if } x = \{\emptyset\}_M \text{ then } \emptyset \text{ else } x))$

lemma *swap-zero-one-involution[simp]*: *swap-zero-one* (*swap-zero-one* x) = x
 $\langle proof \rangle$

lemma *bij-swap-zero-one*: *bij* *swap-zero-one*
 $\langle proof \rangle$

lemma *swap-zero-one-mem*: $a \varepsilon \text{ swap-zero-one } b \longleftrightarrow (b \neq \{\emptyset\}_M \wedge a \varepsilon b) \vee (a = \emptyset \wedge b = \emptyset)$
 $\langle proof \rangle$

lemma *swap-zero-one-perm-model*: *permutation-model* (ε) *swap-zero-one*
 $\langle proof \rangle$

interpretation *swap*: *permutation-model* (ε) *swap-zero-one*
 $\langle proof \rangle$

notation *swap.perm-mem* (**infix** $\langle \varepsilon^s \rangle$ 50)

interpretation *swap*: *L-setext-empty-power-union-repl* (ε^s)
 $\langle proof \rangle$

notation *swap.emptysetM* ($\langle \emptyset^s \rangle$)

notation *swap.setsucM* ($\langle \text{suc}^s \rangle$)

— $\{\emptyset\}$ is the empty set in the permuted model

lemma *one-swap*: $\neg (a \varepsilon^s \{\emptyset\}_M)$
 $\langle proof \rangle$

lemma *swap-empty*: $\emptyset^s = \{\emptyset\}_M$
 $\langle proof \rangle$

lemma *zero-swap-mem-iff*: $a \varepsilon^s \emptyset \longleftrightarrow a = \emptyset$
 $\langle proof \rangle$

Since $\emptyset \varepsilon^s \emptyset$, the permuted model is not regular.

theorem *not-reg-swap-zero-one*: $\neg (L\text{-reg } (\varepsilon^s))$
 $\langle proof \rangle$

In order to show that the model with membership ε^s satisfies *L-setind* (*L-ts* resp.) if the original model does, we first show how the modified successor behaves.

lemma *swap-mem-mem*: $\neg (x = \emptyset \wedge y = \emptyset) \implies x \varepsilon^s y \implies x \varepsilon y$
 $\langle proof \rangle$

lemma *swap.setsuc-1-y*: **assumes** $y \neq \emptyset$ **shows** $\text{suc}^s (\{\emptyset\}_M) y = \{y\}_M$
 $\langle proof \rangle$

lemma *swap.setsuc-0-y*: **assumes** $y \neq \emptyset$ **shows** $\text{suc}^s \emptyset y = \{\emptyset, y\}_M$

<proof>

lemma *swap-setsuc-0-0*: $\text{suc}^s \emptyset = \emptyset$
<proof>

lemma *swap-setsuc-1-0*: $\text{suc}^s (\{\emptyset\}_M) = \emptyset$
<proof>

lemma *swap-setsuc-1-1*: $\text{suc}^s (\{\emptyset\}_M) (\{\emptyset\}_M) = \{\{\emptyset\}_M\}_M$
<proof>

lemma *setsuc-setsuc'*: **assumes** $x \neq \emptyset \wedge x \neq \{\emptyset\}_M$
shows $\text{suc}^s x y = \text{suc } x y$
<proof>

theorem *setind-swap-setind*: $L\text{-setind } (\varepsilon) \longrightarrow L\text{-setind } (\varepsilon^s)$
<proof>

lemma *ts-swap-ts*: $L\text{-ts } (\varepsilon) \longrightarrow L\text{-ts } (\varepsilon^s)$
<proof>

end

theorem *not-reg-model*: **assumes** $L\text{-setext-empty-power-union-repl } (\text{mem} :: 'a \Rightarrow 'a \Rightarrow \text{bool}) \text{ } L\text{-setind } \text{mem } L\text{-ts } \text{mem}$
shows $\neg (\forall (m :: 'a \Rightarrow 'a \Rightarrow \text{bool}). L\text{-setext-empty-power-union-repl } m \wedge L\text{-setind } m \wedge L\text{-ts } m \longrightarrow L\text{-reg } m)$
<proof>

end

2.4 Independence of transitive superset

theory *Not-ts-model*
imports *Permutation-models*
begin

We explore the model defined by the permutation transposing n and $\{n+1\}_M$ for each natural number n . The membership in the model is not well-founded since there is an infinite decreasing chain $\dots 2 \varepsilon^f 1 \varepsilon^f 0$.

Despite that if the native membership satisfies regularity and finiteness, the permuted model also satisfies regularity and finiteness.

Consequently, 0 (the empty set) has not transitive superset, since it could be infinite if it existed. As a corollary, the permutation model does not satisfy schema of regularity (or, equivalently, epsilon induction) since it entails transitive superset axiom.

The construction is given in Rieger in [6] and also in [4] who underscore

that the model is recursive. The statement that regularity schema does not follow from regularity was also proved in [11].

context *L-setext-empty-power-union-repl-reg*

begin

fun *regperm* :: 'a $\Rightarrow$ 'a **where**
 regperm a = (if *natM* a then {*setsucM* a a}_M else
 if ($\exists$ b. *natM* b $\wedge$ a = {*setsucM* b b}_M) then THE b. a = {*setsucM*
 b b}_M else a)

lemma *two-natM*: *natM* (*setsucM* ({ $\emptyset$ }_M)($\{\emptyset\}$ _M))
 <proof>

lemma *suc-sing-not-nat*: $\neg$ *natM* ({*setsucM* b b}_M)
 <proof>

lemma *regperm-image-nat*: **assumes** *natM* a
shows *regperm* a = {*setsucM* a a}_M
 <proof>

lemma *regperm-image-plus*: **assumes** *natM* b
shows *regperm* ({*setsucM* b b}_M) = b
 <proof>

lemma *regperm-image-else*: **assumes** $\neg$ *natM* a $\nexists$ b. *natM* b $\wedge$ a = {*setsucM* b
 b}_M
shows *regperm* a = a
 <proof>

lemma *regperm-bij*: *bij regperm*
 <proof>

lemma *SR-regperm-eq*: *SetRelation* (λ x y. *regperm* x = y)
 <proof>

lemma *SR-regperm-mem*: *SetRelation* (λ x y. x $\in$ *regperm* y)
 <proof>

end

context *L-setext-empty-power-union-repl-reg-fin*

begin

interpretation *perm*: *permutation-model* (ε) *regperm*
 <proof>

abbreviation *fmem* (**infix** ε^f 51) **where**
 fmem x y $\equiv$ *perm.perm-mem* x y

interpretation *fm*: *L-setext-empty-power-union-repl fm*

<proof>

lemma *regperm-mem-nat*: **assumes** *natM a*

shows $u \varepsilon^f a \longleftrightarrow u = (\text{setsucM } a \ a)$

<proof>

lemma *fmem-not-wf*: $\neg (\text{wfp } (\varepsilon^f))$

<proof>

The main meta-theorem. The axiom of the transitive superset does not hold in the model since the transitive closure of $\emptyset$ would be an infinite set.

theorem *regperm-not-ts*: $\neg L\text{-ts } (\varepsilon^f)$

<proof>

theorem *regperm-not-regs*: $\neg L\text{-regs } (\varepsilon^f)$

<proof>

lemma *regperm-reg*:

shows *L-setext-empty-power-union-repl-reg* (ε^f)

<proof>

lemma *perm-empty*: $\text{fm.emptysetM} = \{\{\emptyset\}_M\}_M$

<proof>

lemma *perm-one*: $\text{fm.binunionM } \text{fm.emptysetM } (\text{fm.singletonM } \text{fm.emptysetM}) = \{\{\{\emptyset\}_M\}_M\}_M$

<proof>

lemma *perm-else-setsuc-else*: **assumes** $\neg \text{natM } u \nabla b. \text{natM } b \wedge u = \{\text{setsucM } b\}_M$

shows $\neg \text{natM } (\text{setsucM } u \ u) \nabla b. \text{natM } b \wedge \text{setsucM } u \ u = \{\text{setsucM } b\}_M$

<proof>

lemma *perm-setsuc*: **assumes** $\neg \text{natM } u \nabla b. \text{natM } b \wedge u = \{\text{setsucM } b\}_M$

shows $\text{fm.setsucM } u \ u = \text{setsucM } u \ u$

<proof>

lemma *perm-fin*: *L-fin* (ε^f)

<proof>

interpretation *find*: *L-setind* (ε^f)

<proof>

end

Summary of results. We have shown that if a type admits a membership relation satisfying certain axioms of finite sets (namely axioms of ZF where *inf* is replaced by *fin*), it does not follow that the membership on the type satisfies the existence of transitive supersets or the regularity schema.

theorem *not-reg-setind-implies-regsch-ts*:
 assumes *L-setext-empty-power-union-repl-reg-fin* (*m* :: '*a* $\Rightarrow$ '*a* $\Rightarrow$ bool)
 shows $\neg (\forall (mem :: 'a \Rightarrow 'a \Rightarrow bool). L-setext-empty-power-union-repl-reg\ mem$
 $\wedge L-setind\ mem \longrightarrow L-regsch\ mem \vee L-ts\ mem)$
 $\langle proof \rangle$
end

References

- [1] S. Baratella and R. Ferro. A theory of sets with the negation of the axiom of infinity. *Mathematical Logic Quarterly*, 39:338–352, 1993.
- [2] L. Běhounek. Nezávislost axiomů ve dvou axiomatikách teorie konečných množin. Ročníková práce (equivalent of bachelor thesis), 1998.
- [3] R. Kaye and T. L. Wong. On interpretations of arithmetic and set theory. *Notre Dame Journal of Formal Logic*, 48(4):497–510, 2007.
- [4] A. Mancini and D. Zambella. A note on recursive models of set theories. *Notre Dame Journal of Formal Logic*, 42(2):109–115, 2001.
- [5] P. Pudlák and A. Sochor. Models of the Alternative Set Theory. *The Journal of Symbolic Logic*, 49(2):570–585, 1984.
- [6] L. S. Rieger. A contribution to Gödel’s axiomatic set theory, I. *Czechoslovak Mathematical Journal*, 3:323–357, 1957.
- [7] A. Sochor. Metamathematics of the alternative set theory I. *Commentationes Mathematicae Universitatis Carolinae*, 20(4):697–722, 1979.
- [8] A. Sochor. Metamathematics of the alternative set theory II. *Commentationes Mathematicae Universitatis Carolinae*, 23(1):55–79, 1982.
- [9] A. Sochor. Metamathematics of the alternative set theory III. *Commentationes Mathematicae Universitatis Carolinae*, 24(1):137–154, 1983.
- [10] P. Vopěnka. *Mathematics in the Alternative Set Theory*. Teubner, Leipzig, 1979.
- [11] P. Vopěnka and P. Hájek. Über die Gültigkeit des Fundierungssaxioms in speziellen Systemen der Mengentheorie. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:235–241, 1963.