Simplistic ZFC Formalization

Sebastian Ullrich

May 10, 2014

Contents

theory *ZFC*
imports *HOL*
begin

1 ZFC

typedecl *set*

axiomatization
member :: *set* ⇒ *set* ⇒ *bool*

notation
member (*op* :) **and**
member ((-/ : -) [*51*, *51*] *50*)

abbreviation *not-member* **where**
not-member x A ≡ ~ (*x* : *A*) — non-membership

notation
not-member (*op* ~:) **and**
not-member ((-/ ~: -) [*51*, *51*] *50*)

notation (*xsymbols*)
member (*op* ∈) **and**
member ((-/ ∈ -) [*51*, *51*] *50*) **and**
not-member (*op* ∉) **and**
not-member ((-/ ∉ -) [*51*, *51*] *50*)

1.1 Zermelo-Fraenkel Axiom System

axiomatization where
extensionality: $\forall z.\ (z \in x \longleftrightarrow z \in y) \Longrightarrow x = y$ **and**
foundation: $\exists y.\ y \in x \Longrightarrow \exists y.\ y \in x \wedge (\forall z.\ \neg(z \in x \wedge z \in y))$ **and**
subset-set: $\exists y.\ \forall z.\ z \in y \longleftrightarrow z \in x \wedge P\ z$ **and**
empty-set: $\exists y.\ \forall x.\ x \notin y$ **and**
pair-set: $\exists y.\ \forall x.\ x \in y \longleftrightarrow x = z_1 \vee x = z_2$ **and**
power-set: $\exists y.\ \forall z.\ z \in y \longleftrightarrow (\forall u.\ u \in z \longrightarrow u \in x)$ **and**
sum-set: $\exists y.\ \forall z.\ z \in y \longleftrightarrow (\exists u.\ z \in u \wedge u \in x)$

definition *empty* :: *set* ({}) **where**
empty ≡ *THE* $y.\ \forall x.\ x \notin y$

axiomatization where
infinity: $\exists w.\ \{\} \in w \wedge (\forall x.\ x \in w \longrightarrow (\exists z.\ z \in w \wedge (\forall u.\ u \in z \longleftrightarrow u \in x \vee u = x)))$ **and**
replacement: $P\ x\ y \wedge P\ x\ z \longrightarrow y = z \Longrightarrow \exists u.\ (\forall w_1.\ w_1 \in u \longleftrightarrow (\exists w_2.\ w_2 \in a \wedge P\ w_2\ w_1))$

lemma *empty*[*simp*]: $x \notin empty$
proof−
  have $\exists!y.\ \forall x.\ x \notin y$
  proof (*rule ex-ex1I*)
    fix $y\ y'$
    assume $\forall x.\ x \notin y\ \ \forall x.\ x \notin y'$
    thus $y = y'$ **by** −(*rule extensionality*, *simp*)
  qed (*rule empty-set*)
  hence $\forall x.\ x \notin \{\}$
    unfolding *empty-def*
    by (*rule theI′*)
  thus *?thesis* **..**
qed

Let's try to generalize that for the other introduction axioms.

lemma *exAxiomD1*:
  assumes $\exists y.\ \forall x.\ x \in y \longleftrightarrow P\ x$
  shows $\exists!y.\ \forall x.\ x \in y \longleftrightarrow P\ x$
using *assms*
by (*auto intro*:*extensionality*)

lemma *exAxiomD2*:
  assumes $\exists y.\ \forall x.\ x \in y \longleftrightarrow P\ x$
  shows $x \in (THE\ y.\ \forall x.\ x \in y \longleftrightarrow P\ x) \longleftrightarrow P\ x$
apply (*rule spec*[*of - x*])
by (*rule theI′*[*OF assms*[*THEN exAxiomD1*]])

lemma *exAxiomD3*:
  assumes $\exists y.\ \forall x.\ x \in y \longleftrightarrow P\ x\ \ x \in (THE\ y.\ \forall x.\ x \in y \longleftrightarrow P\ x)$
  shows $P\ x$
using *assms exAxiomD2*
by *auto*

lemma *empty′*: $x \notin empty$
apply (*rule exAxiomD2*[*of* λ-. *False*, *simplified*, *folded empty-def*])
by (*rule empty-set*)

lemma[*simp*]: $(\forall x.\ x \notin y) \longleftrightarrow y = \{\}$
by (*auto intro*:*extensionality*)

definition *pair* :: *set* ⇒ *set* ⇒ *set* ({-, -}) **where**
  $pair\ z_1\ z_2 \equiv THE\ y.\ \forall x.\ x \in y \longleftrightarrow x = z_1 \lor x = z_2$

definition *singleton* :: *set* ⇒ *set* ({-}) **where**
  $singleton\ x \equiv \{x, x\}$

definition *sum* :: *set* ⇒ *set* **where**
  $sum\ x \equiv THE\ y.\ \forall z.\ z \in y \longleftrightarrow (\exists u.\ z \in u \land u \in x)$

definition *subset* :: $(set \Rightarrow bool) \Rightarrow set \Rightarrow set$ **where**
$subset\ P\ x \equiv THE\ y.\ \forall z.\ z \in y \longleftrightarrow z \in x \wedge P\ z$

syntax
subset :: $pttrn => set \Rightarrow bool => set$ $((1\{\text{-} \in \text{-}./\ \text{-}\}))$
translations
$\{z \in x.\ P\} == subset\ (\%z.\ P)\ x$

definition *Pow* :: $set \Rightarrow set$ **where**
$Pow\ x \equiv THE\ y.\ \forall z.\ z \in y \longleftrightarrow (\forall u.\ u \in z \longrightarrow u \in x)$

lemma *pair*[*simp*]: $x \in \{z_1, z_2\} \longleftrightarrow x = z_1 \vee x = z_2$
by (*rule exAxiomD2*[*of* $\lambda x.\ x = z_1 \vee x = z_2$, *simplified*, *folded pair-def*]) (*rule pair-set*)

lemma *singleton*[*simp*]: $x \in \{y\} \longleftrightarrow x = y$
by $-$(*unfold singleton-def*, *simp*)

lemma *sum*[*simp*]: $z \in sum\ x \longleftrightarrow (\exists u.\ z \in u \wedge u \in x)$
by (*rule exAxiomD2*[*of* $\lambda z.\ \exists u.\ z \in u \wedge u \in x$, *simplified*, *folded sum-def*]) (*rule sum-set*)

lemma *subset*[*simp*]: $z \in \{z \in x.\ P\ z\} \longleftrightarrow z \in x \wedge P\ z$
by (*rule exAxiomD2*[*of* $\lambda z.\ z \in x \wedge P\ z$, *simplified*, *folded subset-def*]) (*rule subset-set*)

lemma *Pow*[*simp*]: $z \in Pow\ x \longleftrightarrow (\forall u.\ u \in z \longrightarrow u \in x)$
by (*rule exAxiomD2*[*of* $\lambda z.\ \forall u.\ u \in z \longrightarrow u \in x$, *simplified*, *folded Pow-def*]) (*rule power-set*)

1.2 Lemmas on Unions and Intersubsections

definition *union* :: $set \Rightarrow set \Rightarrow set$ (**infixl** $\cup$ *65*) **where**
$union\ x\ y \equiv sum\ (pair\ x\ y)$

lemma *union*[*simp*]: $z \in a \cup b \longleftrightarrow z \in a \vee z \in b$
by (*auto simp*:*union-def*)

lemma *union-script*: $\exists y.\ \forall z.\ z \in y \longleftrightarrow z \in a \vee z \in b$
by (*rule exI*[*of* - $a \cup b$]) *simp*

definition *intersect* :: $set \Rightarrow set \Rightarrow set$ (**infixl** $\cap$ *70*) **where**
$a \cap b \equiv \{x \in a.\ x \in b\}$

lemma *intersect*[*simp*]: $z \in a \cap b \longleftrightarrow z \in a \wedge z \in b$
by (*simp add*:*intersect-def*)

lemma *intersect-script*: $\exists y.\ \forall z.\ z \in y \longleftrightarrow z \in a \wedge z \in b$

by (*rule subset-set*)

definition *Intersect* :: (*set* $\Rightarrow$ *bool*) $\Rightarrow$ *set* ($\bigcap$ - [*1000*] *999*) **where**
 $\bigcap P \equiv THE\ y.\ \forall z.\ z \in y \longleftrightarrow (\forall u.\ P\ u \longrightarrow z \in u)$

lemma *Intersect*[*simp*]: $\exists z.\ P\ z \Longrightarrow z \in \bigcap P \longleftrightarrow (\forall u.\ P\ u \longrightarrow z \in u)$
proof (*rule exAxiomD2*[*of* $\lambda z.\ \forall u.\ P\ u \longrightarrow z \in u$, *simplified*, *folded Intersect-def*])
 assume $\exists z.\ P\ z$
 then obtain z **where**[*simp*]: $P\ z$ **..**
 let $?y = \{x \in z.\ \forall u.\ P\ u \longrightarrow x \in u\}$
 have $\forall x.\ x \in ?y \longleftrightarrow (\forall u.\ P\ u \longrightarrow x \in u)$ **by** *auto*
 thus $\exists y.\ \forall x.\ x \in y \longleftrightarrow (\forall u.\ P\ u \longrightarrow x \in u)$**..**
qed

definition *Union* :: *set* $\Rightarrow$ *set* ($\bigcup$ - [*1000*] *999*) **where**
 $\bigcup a \equiv THE\ y.\ \forall z.\ z \in y \longleftrightarrow (\exists u.\ z \in u \wedge u \in a)$

lemma *Union*[*simp*]: $z \in \bigcup a \longleftrightarrow (\exists u.\ z \in u \wedge u \in a)$
by (*rule exAxiomD2*[*of* $\lambda z.\ \exists u.\ z \in u \wedge u \in a$, *simplified*, *folded Union-def*]) (*rule sum-set*)

1.3 Ordered Pairs

definition *ordered-pair* :: *set* $\Rightarrow$ *set* $\Rightarrow$ *set* ($\langle$-,-$\rangle$) **where**
 $\langle a,b\rangle \equiv \{\{a\}, \{a,b\}\}$

lemma *intersect-singleton*[*simp*]: $x \cap \{y\} = (if\ y \in x\ then\ \{y\}\ else\ \{\})$
by (*auto intro*:*extensionality*)

lemma *empty-singleton-neq*[*simp*]: $\{x\} \neq \{\}$
proof
 assume *assm*: $\{x\} = \{\}$
 have $x \notin \{\}$ **by** *simp*
 with *assm* **have** $x \notin \{x\}$ **by** *simp*
 thus *False* **by** *simp*
qed

lemma *singleton-eqD*[*dest!*]: $\{x\} = \{y\} \Longrightarrow x = y$
by (*drule arg-cong*[*of - -* $\lambda z.\ y \in z$]) *simp*

lemma *singleton-pair-eqD*[*dest!*]:
 assumes $\{x\} = \{y, z\}$
 shows $x = y \wedge y = z$
proof−
 from *assms* **have** $y \in \{x\} \longleftrightarrow y \in \{y, z\}$ **by** *simp*
 hence $x = y$ **by** *simp*

from *assms* **have** $z \in \{x\} \longleftrightarrow z \in \{y, z\}$ **by** *simp*
hence $x = z$ **by** *simp*
with ‹$x = y$› **show** *?thesis* **by** *simp*
qed

lemma *singleton-pair-eqD'*[*dest*!]:
assumes $\{y, z\} = \{x\}$
shows $x = y \wedge y = z$
using *assms*[*symmetric*] **by** (*rule singleton-pair-eqD*)

lemma *pair-singleton*[*simp*]: $\{x, x\} = \{x\}$
unfolding *singleton-def* **..**

lemma *pair-eq-fstD*[*dest*!]:
assumes $\{x,y\} = \{x,z\}$
shows $y = z$
using *assms*
proof (*cases* $x = y$)
case *False*
from *assms* **have** $y \in \{x,y\} \longleftrightarrow y \in \{x,z\}$ **by** *simp*
with *False* **show** *?thesis* **by** *simp*
qed *auto*

lemma *ordered-pair-eq*[*simp*]: $\langle x_1,x_2\rangle = \langle y_1,y_2\rangle \longleftrightarrow x_1 = y_1 \wedge x_2 = y_2$
proof
assume *assm*: $\langle x_1,x_2\rangle = \langle y_1,y_2\rangle$
hence $\{x_1\} \in \langle x_1,x_2\rangle \longleftrightarrow \{x_1\} \in \langle y_1,y_2\rangle$ **by** *simp*
hence[*simp*]: $x_1 = y_1$ **by** (*auto simp*:*ordered-pair-def*)
show $x_1 = y_1 \wedge x_2 = y_2$ **using** *assm*
proof (*cases* $x_2 = x_1$)
case *False*
from *assm* **have** $\{x_1,x_2\} \in \langle x_1,x_2\rangle \longleftrightarrow \{x_1,x_2\} \in \langle y_1,y_2\rangle$ **by** *simp*
with *False* **show** *?thesis* **by** (*auto simp*:*ordered-pair-def*)
qed (*auto simp*:*ordered-pair-def*)
qed *simp*

1.4 Relations and Functions

definition $rel\ r \equiv \forall x.\ x \in r \longrightarrow (\exists x_1\ x_2.\ x = \langle x_1,x_2\rangle)$
definition $rel''\ r\ a\ b \equiv rel\ r \wedge (\forall x_1\ x_2.\ \langle x_1, x_2\rangle \in r \longrightarrow x_1 \in a \wedge x_2 \in b)$
definition $rel'\ r\ s \equiv rel''\ r\ s\ s$
definition $func\ r \equiv rel\ r \wedge (\forall x\ y_1\ y_2.\ \langle x,y_1\rangle \in r \wedge \langle x,y_2\rangle \in r \longrightarrow y_1 = y_2)$
definition $func'\ f\ a\ b \equiv func\ f \wedge rel''\ f\ a\ b$

1.4.1 Existence Proofs

definition $singletons\ a \equiv \{b \in Pow\ a.\ \exists x.\ b = \{x\}\}$

lemma *singletons*[*simp*]: $b \in singletons\ a \longleftrightarrow (\exists x.\ b = \{x\} \wedge x \in a)$
by (*auto simp*:*singletons-def*)

definition *pairs* $a\ b \equiv \{c \in Pow\ (a \cup b).\ \exists\, x\ y.\ c = \{x,\ y\} \wedge x \in a \wedge y \in b\}$

lemma *pairs-correct*[*simp*]: $c \in pairs\ a\ b \longleftrightarrow (\exists\, x\ y.\ c = \{x,\ y\} \wedge x \in a \wedge y \in b)$
by (*auto simp*:*pairs-def*)

definition *ordered-pairs* $a\ b \equiv \{c \in Pow\ (Pow\ a \cup Pow\ (a \cup b)).\ \exists\, x\ y.\ c = \langle x,y\rangle \wedge x \in a \wedge y \in b\}$

lemma *ordered-pairs*[*simp*]: $c \in ordered\text{-}pairs\ a\ b \longleftrightarrow (\exists\, x\ y.\ c = \langle x,y\rangle \wedge x \in a \wedge y \in b)$
by (*auto simp add*:*ordered-pairs-def ordered-pair-def*)

definition *rels* $a\ b \equiv \{r \in Pow\ (ordered\text{-}pairs\ a\ b).\ rel\ r\}$

lemma *rels*[*simp*]: $r \in rels\ a\ b \longleftrightarrow rel''\ r\ a\ b$
by (*auto simp*:*rels-def rel-def rel''-def*)

definition *funcs* $a\ b \equiv \{f \in rels\ a\ b.\ func\ f\}$

lemma *funcs*[*simp*]: $f \in funcs\ a\ b \longleftrightarrow func'\ f\ a\ b$
by (*auto simp*:*funcs-def func'-def func-def*)

1.5 Natural Numbers

definition *succ* :: *set* ⇒ *set* (-^{+} [*1000*] *999*) **where**
$a^{+} \equiv a \cup \{a\}$

definition *zero* :: *set* (*0*) **where** $0 \equiv \{\}$

definition *Ded* $a \equiv 0 \in a \wedge (\forall\, x.\ x \in a \longrightarrow x^{+} \in a)$

lemma *icanhazded*: $\exists\, a.\ Ded\ a$
proof−
  from *infinity* **obtain** *a* **where** *inf*: $\{\} \in a$
    $\forall\, x.\ x \in a \longrightarrow (\exists\, z.\ z \in a \wedge (\forall\, u.\ (u \in z) = (u \in x \vee u = x)))$ **by** *auto*
  have $\forall\, x.\ x \in a \longrightarrow x^{+} \in a$
  proof (*rule*, *rule*)
    fix *x*
    assume $x \in a$
    with *inf* **obtain** *z* **where** *z*: $z \in a\ \forall\, u.\ u \in z \longleftrightarrow u \in x \vee u = x$ **by** *auto*
    from *this*(*2*) **have**[*simp*]: $z = x \cup \{x\}$
    by (*auto intro*:*extensionality*)
    with *z*(*1*) **show** $x^{+} \in a$ **by** (*auto simp*:*succ-def*)
  qed
  with *inf*(*1*) **show** *?thesis* **by** (*auto simp add*:*Ded-def zero-def*)

qed

definition *nats* :: *set* ($\mathbb{N}$) **where** $\mathbb{N} \equiv \bigcap Ded$

lemma *nats*: $n \in \mathbb{N} \longleftrightarrow (\forall a.\ Ded\ a \longrightarrow n \in a)$
unfolding *nats-def*
by (*rule Intersect*) (*rule icanhazded*)

1.5.1 Peano's Axioms

lemma *ax-zero*[*simp*]: $0 \in \mathbb{N}$
by (*simp add*:*nats Ded-def*)

lemma *ax-succ*[*simp*]: $n \in \mathbb{N} \Longrightarrow n^+ \in \mathbb{N}$
by (*simp add*:*nats Ded-def*)

lemma *nonempty-member*[*simp*]: $x \neq \{\} \longleftrightarrow (\exists y.\ y \in x)$
by (*rule ccontr*) *simp*

lemma *union-nonempty*[*simp*]: $x \neq \{\} \lor y \neq \{\} \Longrightarrow x \cup y \neq \{\}$
by *auto*

lemma *ax-succ-neq-zero*[*simp*]: $n \in \mathbb{N} \Longrightarrow n^+ \neq 0$
by (*simp add*:*succ-def zero-def*)

lemma *ax-succ-inj*:
  assumes $n \in \mathbb{N}$ $m \in \mathbb{N}$ $n^+ = m^+$
  shows $n = m$
proof−
  from *assms*(*3*) **have** $m \in n \cup \{n\}$ **by** (*simp add*:*succ-def*)
  hence *m*: $n = m \lor m \in n$ **by** *auto*
  from *assms*(*3*)[*symmetric*] **have** $n \in m \cup \{m\}$ **by** (*simp add*:*succ-def*)
  hence *n*: $n = m \lor n \in m$ **by** *auto*

  from *n m foundation*[*of* $\{n,m\}$]
  show *?thesis* **by** *auto*
qed

definition *subseteq* :: *set* ⇒ *set* ⇒ *bool* ((- ⊆ -) [*51*,*51*] *50*) **where**
  $x \subseteq y \equiv \forall z.\ z \in x \longrightarrow z \in y$

lemma *empty-subseteq*[*simp*]: $\{\} \subseteq x$
by (*simp add*:*subseteq-def*)

lemma *singleton-subseteq*[*simp*]: $\{x\} \subseteq y \longleftrightarrow x \in y$
by (*simp add*:*subseteq-def*)

lemma *subseteq-trans*[*trans*]: $[\![x \subseteq y;\ y \subseteq z]\!] \Longrightarrow x \subseteq z$
by (*simp add*:*subseteq-def*)

lemma *subseteq-member*[*trans*]: $[\![x \in y;\ y \subseteq z]\!] \Longrightarrow x \in z$
by (*simp add*:*subseteq-def*)

lemma *subseteqI*[*intro*]: $(\bigwedge z.\ z \in x \Longrightarrow z \in y) \Longrightarrow x \subseteq y$
by (*simp add*:*subseteq-def*)

lemma *subseteq-unionI*[*intro*!]: $x \subseteq y \vee x \subseteq z \Longrightarrow x \subseteq y \cup z$
by (*auto simp add*:*subseteq-def*)

lemma *union-subseteqI*[*simp*]: $x \cup y \subseteq z \longleftrightarrow x \subseteq z \wedge y \subseteq z$
by (*auto simp add*:*subseteq-def*)

lemma *ax-induct*: $[\![0 \in x;\ \bigwedge y.\ y \in x \Longrightarrow y^+ \in x]\!] \Longrightarrow \mathbb{N} \subseteq x$
by (*simp add*:*subseteq-def nats Ded-def*)

1.5.2 Set Properties of $\mathbb{N}$

lemma[*simp*]: $0 \in 0^+$
by (*simp add*:*succ-def*)

lemma[*simp*]: $0 \in y \Longrightarrow 0 \in y^+$
by (*simp add*:*succ-def*)

lemma
  assumes $n \in \mathbb{N}$ $n \neq 0$
  shows $0 \in n$
proof−
  let $?x = \{n \in \mathbb{N}.\ 0 \in n\} \cup \{0\}$
  note *assms*(*1*)
  also
  have $\mathbb{N} \subseteq ?x$ **by** (*rule ax-induct*) *auto*
  finally have $n \in ?x$ **.**
  with *assms*(*2*) **show** *?thesis* **by** *auto*
qed

lemma *nat-induct*[*case-names Zero Succ*[*hyps IH*], *consumes 1*]:
  assumes $n \in \mathbb{N}$
  and $P\ 0$ $\bigwedge n.\ [\![n \in \mathbb{N};\ P\ n]\!] \Longrightarrow P\ (n^+)$
  shows $P\ n$
proof−
  let $?x = \{n \in \mathbb{N}.\ P\ n\}$
  note $\langle n \in \mathbb{N}\rangle$
  also have $\mathbb{N} \subseteq ?x$ **by** (*rule ax-induct*, *simp-all add*:*assms*(*2*,*3*))
  finally have $n \in ?x$ **.**
  thus $P\ n$ **by** *simp*
qed

1.5.3 Transitive Sets

definition *trans* $a \equiv \forall x.\ x \in a \longrightarrow x \subseteq a$

lemma *trans'*: $trans\ a \longleftrightarrow (\forall x\ y.\ x \in a \wedge y \in x \longrightarrow y \in a)$
by (*auto simp add*:*trans-def subseteq-member*)

lemma *succ-subseteq*[*simp*]: $n \subseteq n^+$
by (*auto simp add*:*succ-def*)

lemma *succE*:
  assumes $n \in m^+$
  obtains $n \in m \mid n = m$
proof (*cases* $n \in m$)
  case *False*
  assume $n = m \Longrightarrow thesis$
  with *False assms*(*1*) **show** *thesis* **by** (*simp add*:*succ-def*)
qed *simp*

lemma[*simp*]: $n \notin 0$
by (*simp add*:*zero-def*)

lemma $n \in \mathbb{N} \Longrightarrow trans\ n$
proof (*induct rule*:*nat-induct*)
  case *Zero*
  show *?case* **by** (*simp add*:*trans-def*)
next
  case (*Succ n*)
  show *?case*
  unfolding *trans-def*
  proof (*rule*, *rule*)
    fix *x*
    assume $x \in n^+$
    thus $x \subseteq n^+$
    proof (*cases x n rule*:*succE*)
      case *1*
      with ⟨*trans n*⟩ **have** $x \subseteq n$ **by** (*simp add*:*trans-def*)
      also have $n \subseteq n^+$ **by** *simp*
      finally show *?thesis* **.**
    next
      case *2*
      thus *?thesis* **by** (*auto simp add*:*succ-def*)
    qed
  qed
qed

lemma *trans* $\mathbb{N}$
unfolding *trans-def*
proof (*rule*, *rule*)
  fix *n*

```
  assume n ∈ ℕ
  thus n ⊆ ℕ
  by (rule nat-induct) (simp-all add:zero-def succ-def)
qed
```

1.5.4 The order relation on ℕ

```
definition trans-rel r ≡ ∀ x y z. ⟨x,y⟩ ∈ r ∧ ⟨y,z⟩ ∈ r ⟶ ⟨x,z⟩ ∈ r

lemma trans-relD:
  assumes trans-rel r ⟨x,y⟩ ∈ r ⟨y,z⟩ ∈ r
  shows ⟨x,z⟩ ∈ r
proof−
  from assms(1) have ⟨x,y⟩ ∈ r ∧ ⟨y,z⟩ ∈ r ⟶ ⟨x,z⟩ ∈ r
    unfolding trans-rel-def
    by −(erule allE)+
  with assms(2,3) show ?thesis by simp
qed

lemma
  assumes trans-rel r ⋀n. ⟨n,n⁺⟩ ∈ r m ∈ ℕ
  shows n ∈ m ⟶ ⟨n,m⟩ ∈ r
using assms(3) proof (induct m rule:nat-induct)
  case (Succ m)
  show ?case
  proof
    assume n ∈ m⁺
    thus ⟨n,m⁺⟩ ∈ r
    proof (cases n m rule:succE)
      case 1
      show ?thesis proof (rule trans-relD[OF assms(1)])
        from 1 show ⟨n,m⟩ ∈ r by (rule Succ.IH[THEN mp])
        show ⟨m,m⁺⟩ ∈ r by (rule assms(2))
      qed
    qed (simp add:assms(2))
  qed
qed simp
```

1.5.5 Set Properties of ℕ (II)

```
definition less (- < - [51, 51] 50) where n < m ≡ n ∈ m

lemma trans-nat: ⟦n ∈ ℕ; m ∈ n⟧ ⟹ m ∈ ℕ
proof (induct rule:nat-induct)
  case (Succ n)
  from this(3) show ?case by (cases m n rule:succE) (auto intro:Succ)
qed simp

lemma n ∈ ℕ ⟹ n = {m ∈ ℕ . m < n}
unfolding less-def
```

by (*rule extensionality*) (*auto intro*:*trans-nat*)

end

theory *ModalLogic*
imports *Main*
begin

2 Modal Logic

2.1 Definition

datatype *'a PModFml*
= *V 'a*
| *Not 'a PModFml* ($\neg_M$- [*140*] *140*)
| *And 'a PModFml 'a PModFml* (**infixr** $\wedge_M$ *125*)
| *Box 'a PModFml* ($\Box$- [*140*] *140*)

definition *Or* :: *'a PModFml* ⇒ *'a PModFml* ⇒ *'a PModFml* (**infixr** $\vee_M$ *130*)
where
$x \vee_M y \equiv \neg_M(\neg_M x \wedge_M \neg_M y)$

definition *Implies* :: *'a PModFml* ⇒ *'a PModFml* ⇒ *'a PModFml* (**infixr** $\longrightarrow_M$ *125*) **where**
$x \longrightarrow_M y \equiv \neg_M x \vee_M y$

definition *Iff* :: *'a PModFml* ⇒ *'a PModFml* ⇒ *'a PModFml* (**infixr** $\longleftrightarrow_M$ *110*)
where
$x \longleftrightarrow_M y \equiv (x \longrightarrow_M y) \wedge_M (y \longrightarrow_M x)$

definition *Diamond* :: *'a PModFml* ⇒ *'a PModFml* ($\Diamond$- [*140*] *140*) **where**
$\Diamond x \equiv \neg_M \Box \neg_M x$

definition *Mod-True* :: *'a PModFml* ($True_M$) **where**
$True_M \equiv$ *V undefined* $\vee_M \neg_M$ *V undefined*

typedef *'g KripkeFrame* = {(*G* :: *'g set*, *R* :: *'g rel*). *Field R* ⊆ *G*} **by** *auto*

abbreviation *G Fr* ≡ *fst* (*Rep-KripkeFrame Fr*)
abbreviation *R Fr* ≡ *snd* (*Rep-KripkeFrame Fr*)

lemma *Frame-wf*: *Field* (*R Fr*) ⊆ *G Fr*
using *Rep-KripkeFrame*[*of Fr*] **by** *auto*

lemma[*simp*]:
assumes (*g*,*h*) ∈ *R Fr*

shows $g \in G\ Fr\ h \in G\ Fr$
proof−
from *assms* **have** $g \in Domain\ (R\ Fr)$ **by** *auto*
also from *Frame-wf*[*of Fr*] **have** $Domain\ (R\ Fr) \subseteq G\ Fr$ **by** (*simp add*:*Field-def*)
finally show $g \in G\ Fr$ **.**
from *assms* **have** $h \in Range\ (R\ Fr)$ **by** *auto*
also from *Frame-wf*[*of Fr*] **have** $Range\ (R\ Fr) \subseteq G\ Fr$ **by** (*simp add*:*Field-def*)
finally show $h \in G\ Fr$ **.**
qed

type-synonym $('g, 'a)\ KripkeStruct = 'g\ KripkeFrame \times (['g, 'a] \Rightarrow bool)$

abbreviation $Frame\ \mathcal{K} \equiv fst\ \mathcal{K}$
abbreviation $v\ \mathcal{K} \equiv snd\ \mathcal{K}$
abbreviation $G'\ \mathcal{K} \equiv G\ (Frame\ \mathcal{K})$
abbreviation $R'\ \mathcal{K} \equiv R\ (Frame\ \mathcal{K})$

fun *eval* :: $[('g, 'a)\ KripkeStruct, 'g, 'a\ PModFml] \Rightarrow bool\ (\langle\text{-},\text{-}\rangle \models \text{-}\ [0,0,51]\ 50)$
where
$\langle\mathcal{K},g\rangle \models (V\ var) = (v\ \mathcal{K})\ g\ var$
$|\ \langle\mathcal{K},g\rangle \models \neg_M f = (\neg\ \langle\mathcal{K},g\rangle \models f)$
$|\ \langle\mathcal{K},g\rangle \models f1 \wedge_M f2 = (\langle\mathcal{K},g\rangle \models f1 \wedge \langle\mathcal{K},g\rangle \models f2)$
$|\ \langle\mathcal{K},g\rangle \models \Box f = (\forall h.\ (g,h) \in R'\ \mathcal{K} \longrightarrow \langle\mathcal{K},h\rangle \models f)$

lemma *eval-or*[*simp*]: $\langle\mathcal{K},g\rangle \models f1 \vee_M f2 \longleftrightarrow \langle\mathcal{K},g\rangle \models f1 \vee \langle\mathcal{K},g\rangle \models f2$ **by** (*simp add*:*Or-def*)
lemma *eval-implies*[*simp*]: $\langle\mathcal{K},g\rangle \models f1 \longrightarrow_M f2 \longleftrightarrow \langle\mathcal{K},g\rangle \models f1 \longrightarrow \langle\mathcal{K},g\rangle \models f2$ **by** (*simp add*:*Implies-def*)
lemma *eval-iff*[*simp*]: $\langle\mathcal{K},g\rangle \models f1 \longleftrightarrow_M f2 \longleftrightarrow (\langle\mathcal{K},g\rangle \models f1) = (\langle\mathcal{K},g\rangle \models f2)$ **by** (*auto simp add*:*Iff-def*)
lemma *eval-diamond*[*simp*]: $\langle\mathcal{K},g\rangle \models \Diamond f \longleftrightarrow (\exists h.\ (g,h) \in R'\ \mathcal{K} \wedge \langle\mathcal{K},h\rangle \models f)$ **by** (*simp add*:*Diamond-def*)
lemma *eval-true*[*simp*]: $\langle\mathcal{K},g\rangle \models True_M$ **by** (*simp add*:*Mod-True-def*)

lemmas *eval-impliesI* = *eval-implies*[*THEN iffD2*, *rule-format*]
lemmas *eval-boxD* = *eval.simps*(*4*)[*THEN iffD1*, *rule-format*]

abbreviation *global-eval* :: $[('g, 'a)\ KripkeStruct, 'a\ PModFml] \Rightarrow bool\ (\text{-} \models \text{-}\ [51,51]\ 50)$ **where**
$\mathcal{K} \models F \equiv (\forall g \in G'\ \mathcal{K}.\ \langle\mathcal{K},g\rangle \models F)$

2.2 Tautologies

abbreviation $tautology\ F \equiv \forall(\mathcal{K} :: (nat,\text{-})\ KripkeStruct).\ \mathcal{K} \models F$

lemma *taut1*: $tautology\ (\Box F \longleftrightarrow_M \neg_M \Diamond \neg_M F)$ **by** *auto*
lemma *taut2*: $tautology\ (\Box(P \longrightarrow_M Q) \longrightarrow_M (\Box P \longrightarrow_M \Box Q))$ **by** *simp*
lemma *taut3*: $tautology\ (\Box(P \wedge_M Q) \longleftrightarrow_M (\Box P \wedge_M \Box Q))$ **by** *auto*
lemma *taut4*: $tautology\ (\Diamond(P \vee_M Q) \longleftrightarrow_M (\Diamond P \vee_M \Diamond Q))$ **by** *auto*

lemma *taut5*: *tautology* $((\Box P \vee_M \Box Q) \longrightarrow_M \Box(P \vee_M Q))$ **by** *simp*
lemma *taut6*: *tautology* $(\Diamond(P \wedge_M Q) \longrightarrow_M (\Diamond P \wedge_M \Diamond Q))$ **by** *auto*

2.3 Classes of Kripke Frames

type-synonym $'g$ *KripkeClass* $=$ $'g$ *KripkeFrame set*
abbreviation K :: $'g$ *KripkeClass* **where** $K \equiv$ *UNIV*
abbreviation $T \equiv \{Fr \in K.$ *refl-on* $(G\ Fr)\ (R\ Fr)\}$

lemma T[*simp*]: $Fr \in T \longleftrightarrow (\forall g \in G\ Fr.\ (g,g) \in R\ Fr)$ **using** *Frame-wf*[*of Fr*]
by (*auto simp*:*refl-on-def*)

2.4 Relative Tautologies

definition *CTaut* $C\ F \equiv \forall Fr \in C.\ \forall v.\ \forall g \in G\ Fr.\ \langle (Fr,v),g\rangle \models F$

lemma *pred-def-rewrite*: $P \equiv Q \Longrightarrow Q \Longrightarrow P$ **by** *simp*
lemmas *CTautI* = *CTaut-def*[*THEN pred-def-rewrite*, *rule-format*]

lemma *ttaut1*: *CTaut* $(T$ :: $'g$ *KripkeClass*$)$ $(\Box p \longrightarrow_M p)$
proof (*rule CTautI*)
  fix *Fr* :: $'g$ *KripkeFrame*
  fix v :: $'g \Rightarrow {'a} \Rightarrow bool$
  fix g
  let $?\mathcal{K} = (Fr,v)$
  assume $Fr \in T$ $g \in G\ Fr$
  from *this*(*1*) **have** *refl-on* $(G\ Fr)\ (R\ Fr)$ **by** *simp*
  hence $(g,g) \in R\ Fr$ **using** ‹$g \in G\ Fr$› **by** (*rule refl-onD*)
  hence $(g,g) \in R'\ ?\mathcal{K}$ **by** *simp*
  show $\langle ?\mathcal{K},g\rangle \models (\Box p \longrightarrow_M p)$
  proof (*rule eval-impliesI*)
    assume $\langle ?\mathcal{K},g\rangle \models (\Box p)$
    thus $\langle ?\mathcal{K},g\rangle \models p$ **using** ‹$(g,g) \in R'\ ?\mathcal{K}$› **by** (*rule eval-boxD*)
  qed
qed

lemma *ttaut2*: *CTaut* T $(p \longrightarrow_M \Diamond p)$
by (*auto simp*:*CTaut-def intro*:*refl-onD*)

lemma *ttaut3*: *CTaut* T $(\Box\Box p \longrightarrow_M \Box p)$
by (*rule ttaut1*)

lemma *ttaut4*: *CTaut* T $(\Box\Diamond p \longrightarrow_M \Diamond p)$
by (*rule ttaut1*)

lemma *ttaut5*: *CTaut* T $(\Box p \longrightarrow_M \Diamond\Box p)$
by (*rule ttaut2*)

lemma *ttaut6*: *CTaut T* $(\Diamond p \longrightarrow_M \Diamond\Diamond p)$
by (*rule ttaut2*)

2.5 Modal Logical Consequence

definition *conseq* :: $['a\ PModFml, 'a\ PModFml] \Rightarrow bool$ (- $\models_L$ - [31,31] 30) **where**
$M \models_L F \equiv \forall \mathcal{K} :: (nat,\text{-})\ KripkeStruct.\ \forall g \in G'\ \mathcal{K}.\ \langle\mathcal{K},g\rangle \models M \longrightarrow \langle\mathcal{K},g\rangle \models F$

definition *global-conseq* :: $['a\ PModFml, 'a\ PModFml] \Rightarrow bool$ (- $\models_G$ - [31,31] 30) **where**
$M \models_G F \equiv \forall \mathcal{K} :: (nat,\text{-})\ KripkeStruct.\ \mathcal{K} \models M \longrightarrow \mathcal{K} \models F$

definition *rel-conseq* :: $['a\ PModFml, 'g\ KripkeClass, 'a\ PModFml] \Rightarrow bool$ (- $\models_L$‾ - [31,0,31] 30) **where**
$M \models_L{}^C F \equiv \forall \mathcal{K}.\ Frame\ \mathcal{K} \in C \longrightarrow (\forall g \in G'\ \mathcal{K}.\ \langle\mathcal{K},g\rangle \models M \longrightarrow \langle\mathcal{K},g\rangle \models F)$

definition *global-rel-conseq* :: $['a\ PModFml, 'g\ KripkeClass, 'a\ PModFml] \Rightarrow bool$ (- $\models_G$‾ - [31,0,31] 30) **where**
$M \models_G{}^C F \equiv \forall \mathcal{K}.\ Frame\ \mathcal{K} \in C \longrightarrow \mathcal{K} \models M \longrightarrow \mathcal{K} \models F$

2.6 Model Deduction Theorem

theorem *modal-deduction*: $F_1 \models_L F_2 \longleftrightarrow True_M \models_L (F_1 \longrightarrow_M F_2)$ **by** (*simp add:conseq-def*)

end

theory *ModalCharacterization*
imports *ModalLogic*
begin

3 Correspondence Theory

3.1 What Is It About?

abbreviation *char* $\mathcal{G}\ F \equiv \forall Fr.\ Fr \in \mathcal{G} \longleftrightarrow (\forall v.\ (Fr,v) \models F)$

lemma *char* $(T :: 'g\ KripkeClass)\ (\Box V\ p \longrightarrow_M V\ p)$
proof (*rule*, *rule*)
  fix $Fr :: 'g\ KripkeFrame$
  show $Fr \in T \Longrightarrow \forall v.\ (Fr,\ v) \models \Box V\ p \longrightarrow_M V\ p$ **by** (*auto intro:refl-onD*)
next
  fix $Fr :: 'g\ KripkeFrame$
  assume *asm*: $\forall v.\ (Fr,\ v) \models \Box V\ p \longrightarrow_M V\ p$
  show $Fr \in T$
  proof (*rule ccontr*)
    assume $Fr \notin T$
   then obtain g_0 **where** *g0*: $g_0 \in G\ Fr\ (g_0,g_0) \notin R\ Fr$ **by** (*auto simp:refl-on-def*)
    let $?v0 = \lambda g\ p.\ (g_0,g) \in R\ Fr$

have $\neg \langle (Fr,?v0),g_0\rangle \models \Box V\ p \longrightarrow_M\ V\ p$ **using** *g0(2)* **by** *simp*
with *asm*[*THEN spec*[**where** $x=?v0$], *THEN bspec*[**where** $x=g_0$]] *g0(1)* **show** *False* **by** *simp*
qed
qed

end