FormSys2 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**
  $empty \equiv THE\ y.\ \forall x.\ x \notin y$

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 $x \in (\mathit{THE}\ y.\ \forall x.\ x \in y \longleftrightarrow P\ x) \longleftrightarrow P\ x$
apply (*rule spec*[*of* - *x*])
by (*rule theI'*[*OF assms*[*THEN exAxiomD1*]])

lemma *exAxiomD3*:
  assumes $\exists y.\ \forall x.\ x \in y \longleftrightarrow P\ x\ \ x \in (\mathit{THE}\ y.\ \forall x.\ x \in y \longleftrightarrow P\ x)$
  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**
  $\mathit{pair}\ z_1\ z_2 \equiv \mathit{THE}\ y.\ \forall x.\ x \in y \longleftrightarrow x = z_1 \lor x = z_2$

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

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

definition *subset* :: (*set* ⇒ *bool*) ⇒ *set* ⇒ *set* **where**
subset P x ≡ *THE y*. ∀ *z*. *z* ∈ *y* ⟷ *z* ∈ *x* ∧ *P z*

syntax
subset :: *pttrn* => *set* ⇒ *bool* => *set* ((*1*{- ∈ -./ -}))
translations
{*z* ∈ *x*. *P*} == *subset* (%*z*. *P*) *x*

definition *Pow* :: *set* ⇒ *set* **where**
Pow x ≡ *THE y*. ∀ *z*. *z* ∈ *y* ⟷ (∀ *u*. *u* ∈ *z* ⟶ *u* ∈ *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* ∈ {*y*} ⟷ *x* = *y*
by −(*unfold singleton-def*, *simp*)

lemma *sum*[*simp*]: *z* ∈ *sum x* ⟷ (∃ *u*. *z* ∈ *u* ∧ *u* ∈ *x*)
by (*rule exAxiomD2*[*of* λ*z*. ∃ *u*. *z* ∈ *u* ∧ *u* ∈ *x*, *simplified*, *folded sum-def*]) (*rule sum-set*)

lemma *subset*[*simp*]: *z* ∈ {*z* ∈ *x*. *P z*} ⟷ *z* ∈ *x* ∧ *P z*
by (*rule exAxiomD2*[*of* λ*z*. *z* ∈ *x* ∧ *P z*, *simplified*, *folded subset-def*]) (*rule subset-set*)

lemma *Pow*[*simp*]: *z* ∈ *Pow x* ⟷ (∀ *u*. *u* ∈ *z* ⟶ *u* ∈ *x*)
by (*rule exAxiomD2*[*of* λ*z*. ∀ *u*. *u* ∈ *z* ⟶ *u* ∈ *x*, *simplified*, *folded Pow-def*]) (*rule power-set*)

1.2 Lemmas on Unions and Intersubsections

definition *union* :: *set* ⇒ *set* ⇒ *set* (**infixl** ∪ *65*) **where**
union x y ≡ *sum* (*pair x y*)

lemma *union*[*simp*]: *z* ∈ *a* ∪ *b* ⟷ *z* ∈ *a* ∨ *z* ∈ *b*
by (*auto simp*:*union-def*)

lemma *union-script*: ∃ *y*. ∀ *z*. *z* ∈ *y* ⟷ *z* ∈ *a* ∨ *z* ∈ *b*
by (*rule exI*[*of - a* ∪ *b*]) *simp*

definition *intersect* :: *set* ⇒ *set* ⇒ *set* (**infixl** ∩ *70*) **where**
a ∩ *b* ≡ {*x* ∈ *a*. *x* ∈ *b*}

lemma *intersect*[*simp*]: *z* ∈ *a* ∩ *b* ⟷ *z* ∈ *a* ∧ *z* ∈ *b*
by (*simp add*:*intersect-def*)

lemma *intersect-script*: ∃ *y*. ∀ *z*. *z* ∈ *y* ⟷ *z* ∈ *a* ∧ *z* ∈ *b*

by (*rule subset-set*)

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

definition *Iff* :: *'a PModFml* $\Rightarrow$ *'a PModFml* $\Rightarrow$ *'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* $\Rightarrow$ *'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* $\subseteq$ *G*} **by** *auto*

abbreviation *G Fr* $\equiv$ *fst* (*Rep-KripkeFrame Fr*)
abbreviation *R Fr* $\equiv$ *snd* (*Rep-KripkeFrame Fr*)

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

lemma[*simp*]:
assumes (*g*,*h*) $\in$ *R Fr*

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

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

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

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

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

lemmas $\mathit{eval\text{-}impliesI}[\mathit{intro}] = \mathit{eval\text{-}implies}[\mathit{THEN\ iffD2},\ \mathit{rule\text{-}format}]$
lemmas $\mathit{eval\text{-}boxI}[\mathit{intro}] = \mathit{eval.simps}(4)[\mathit{THEN\ iffD2},\ \mathit{rule\text{-}format}]$
lemmas $\mathit{eval\text{-}boxD}[\mathit{dest}] = \mathit{eval.simps}(4)[\mathit{THEN\ iffD1},\ \mathit{rule\text{-}format}]$

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

2.2 Tautologies

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

lemma $\mathit{taut1}$: $\mathit{tautology}\ (\Box F \longleftrightarrow_M \neg_M \Diamond \neg_M F)$ **by** *auto*
lemma $\mathit{taut2}$: $\mathit{tautology}\ (\Box(P \longrightarrow_M Q) \longrightarrow_M (\Box P \longrightarrow_M \Box Q))$ **by** *simp*
lemma $\mathit{taut3}$: $\mathit{tautology}\ (\Box(P \wedge_M Q) \longleftrightarrow_M (\Box P \wedge_M \Box Q))$ **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* $(G\ Fr)\ (R\ Fr)\}$
abbreviation $S4 \equiv \{Fr \in T.$ *trans* $(R\ Fr)\}$
abbreviation $S5 \equiv \{Fr \in S4.$ *sym* $(R\ Fr)\}$
abbreviation $K4 \equiv \{Fr \in K.$ *trans* $(R\ Fr)\}$
abbreviation $B \equiv \{Fr \in K4.$ *sym* $(R\ Fr)\}$
abbreviation $D \equiv \{Fr \in K.$ *Domain* $(R\ Fr) = G\ Fr\}$

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

definition *rel-conseq* :: $['a\ PModFml, 'g\ KripkeClass, 'a\ PModFml] \Rightarrow bool$ $(\text{-} \models_L\text{\textasciicircum} \text{-}\ [31,0,31]\ 30)$ **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* :: $['a\ PModFml, 'g\ KripkeClass, 'a\ PModFml] \Rightarrow bool$ $(\text{-} \models_G\text{\textasciicircum} \text{-}\ [31,0,31]\ 30)$ **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

lemma *char* (*K4* :: *'g KripkeClass*) ($\Box V\ p \longrightarrow_M\ \Box\Box V\ p$)
proof (*rule*, *rule*)
fix *Fr* :: *'g KripkeFrame*
show $Fr \in K4 \Longrightarrow \forall\, v.\ (Fr,\ v) \models \Box V\ p \longrightarrow_M\ \Box\Box V\ p$ **by** (*auto elim*:*transE*)
next
fix *Fr* :: *'g KripkeFrame*
assume *asm*: $\forall\, v.\ (Fr,\ v) \models \Box V\ p \longrightarrow_M\ \Box\Box V\ p$
show $Fr \in K4$
proof (*rule ccontr*)
assume $Fr \notin K4$
then obtain *x y z* **where** *xyz*: $x \in G\ Fr\ y \in G\ Fr\ z \in G\ Fr\ (x,y) \in R\ Fr\ (y,z) \in R\ Fr\ (x,z) \notin R\ Fr$ **by** (*auto simp*:*trans-def*)
let *?v* $= \lambda g\ p.\ g \neq z$
have $\neg\ \langle (Fr, ?v), x \rangle \models \Box V\ p \longrightarrow_M\ \Box\Box V\ p$ **using** *xyz* **by** *auto*
with *asm*[*THEN spec*[**where** *x*=*?v*], *THEN bspec*[**where** *x*=*x*]] **show** *False* **by** *simp*
qed
qed

3.2 A More General Characterization

no-notation *power* (**infixr** ˆ *80*)
primrec *power* :: *'a rel* ⇒ *nat* ⇒ *'a rel* (**infixr** ˆ *80*) **where**
power-0: *a* ˆ *0* = *Id*
| *power-Suc*: *a* ˆ *Suc n* = *a* ˆ *n O a*

notation (*latex* **output**)
power ((-⁻) [*1000*] *1000*)

primrec *box-n* :: *nat* ⇒ *'a PModFml* ⇒ *'a PModFml* (□⁻- [*0*,*140*] *140*) **where**
$\Box^0\ F = F$
| *box-n* (*Suc n*) $F = \Box^n\Box\ F$

primrec *diamond-n* :: *nat* ⇒ *'a PModFml* ⇒ *'a PModFml* (◇⁻- [*0*,*140*] *140*) **where**
$\Diamond^0\ F = F$
| *diamond-n* (*Suc n*) $F = \Diamond^n\Diamond\ F$

lemma *box-n*[*simp*]:
  assumes $g \in G' \mathcal{K}$
  shows $\langle\mathcal{K},g\rangle \models \Box^n\ F \longleftrightarrow (\forall\, h \in G' \mathcal{K}.\ (g,h) \in (R' \mathcal{K})\hat{}n \longrightarrow \langle\mathcal{K},h\rangle \models F)$
proof (*induction n arbitrary*: F)
  case (*Suc n*)
  show *?case*
  proof (*rule*, *rule*, *rule*)
    fix h
    assume *asm*: $\langle\mathcal{K},g\rangle \models$ *box-n* (*Suc n*) $F\ h \in G' \mathcal{K}\ (g,\ h) \in R' \mathcal{K}$ ^ *Suc n*
    with *Suc.IH*[*THEN iffD1*, *OF asm*(*1*)[*simplified*]] **show** $\langle\mathcal{K},h\rangle \models F$ **by** *auto*
  next
    assume $\forall\, h{\in}G' \mathcal{K}.\ (g,\ h) \in R' \mathcal{K}$ ^ *Suc n* $\longrightarrow \langle\mathcal{K},h\rangle \models F$
    hence $\langle\mathcal{K},g\rangle \models \Box^n\Box F$ **by** − (*rule Suc.IH*[*THEN iffD2*], *auto*)
    thus $\langle\mathcal{K},g\rangle \models$ *box-n* (*Suc n*) F **by** *simp*
  qed
qed (*simp add*:*assms*)

lemmas *box-nI*[*intro*] = *box-n*[*THEN iffD2*, *rule-format*]

lemma *diamond-n*[*simp*]:
  assumes $g \in G' \mathcal{K}$
  shows $\langle\mathcal{K},g\rangle \models \Diamond^n\ F \longleftrightarrow (\exists\, h \in G' \mathcal{K}.\ (g,h) \in (R' \mathcal{K})\hat{}n \wedge \langle\mathcal{K},h\rangle \models F)$
proof (*induction n arbitrary*: F)
  case (*Suc n*)
  show *?case*
  proof
    fix h
    assume *asm*: $\langle\mathcal{K},g\rangle \models$ *diamond-n* (*Suc n*) F
    with *Suc.IH*[*THEN iffD1*, *OF asm*(*1*)[*simplified*]] **show** $\exists\, h{\in}G' \mathcal{K}.\ (g,\ h) \in R' \mathcal{K}$ ^ *Suc n* $\wedge \langle\mathcal{K},h\rangle \models F$ **by** *auto*
  next
    assume $\exists\, h{\in}G' \mathcal{K}.\ (g,\ h) \in R' \mathcal{K}$ ^ *Suc n* $\wedge \langle\mathcal{K},h\rangle \models F$
    hence $\langle\mathcal{K},g\rangle \models \Diamond^n\Diamond F$ **by** − (*rule Suc.IH*[*THEN iffD2*], *auto*)
    thus $\langle\mathcal{K},g\rangle \models$ *diamond-n* (*Suc n*) F **by** *simp*
  qed
qed (*simp add*:*assms*)

lemmas *diamond-nI*[*intro*] = *diamond-n*[*THEN iffD2*, *rule-format*]

3.3 The C Property

abbreviation $C\ m\ n\ j\ k \equiv \{Fr \in K.\ \forall\, w_1\ w_2\ w_3.\ (w_1,w_3) \in R\ Fr$ ^ $m \wedge (w_1,w_2) \in R\ Fr$ ^ $j \longrightarrow (\exists\, w_4.\ (w_3,w_4) \in R\ Fr$ ^ $n \wedge (w_2,w_4) \in R\ Fr$ ^ $k)\}$

lemma $\neg$ *char* (*C 0 0 0 1* :: *'a KripkeClass*) $(\Diamond^0\Box^0\ (V\ p) \longrightarrow_M \Box^0\Diamond^1\ (V\ p))$
proof−
  let *?Fr* = *Abs-KripkeFrame* ({},{})
  have[*simp*]: *G* (*?Fr*) = {} **using** *Abs-KripkeFrame-inverse*[*of* ({},{})] **by** (*metis*

(*no-types*) *Field-empty empty-subsetI fst-conv mem-Collect-eq split-conv*)
  have[*simp*]: R (*?Fr*) = {} **using** *Abs-KripkeFrame-inverse*[*of* ({},{})] **by** (*metis*
(*no-types*) *Field-empty empty-subsetI fst-conv mem-Collect-eq split-conv surjective-pairing*)
  have l: $\neg$ *?Fr* $\in C\ 0\ 0\ 0\ 1$ **by** *simp*
  have r: $\forall v.$ (*?Fr*,$\lambda g\ p.$ *True*) $\models (\Diamond^0\Box^0\ (V\ p) \longrightarrow_M \Box^0\Diamond^1\ (V\ p))$ **by** *simp*
  show *?thesis*
  apply (*rule*, *drule spec*[**where** x= *?Fr*])
  using $l\ r$ **by** *auto*
qed

— Well... maybe this definition shouldn't be taken quite as literally

abbreviation $C'\ m\ n\ j\ k \equiv \{Fr \in K.\ \forall w_1 \in G\ Fr.\ \forall w_2 \in G\ Fr.\ \forall w_3 \in G\ Fr.$
$(w_1,w_3) \in R\ Fr\ \hat{}\ m \wedge (w_1,w_2) \in R\ Fr\ \hat{}\ j \longrightarrow (\exists w_4.\ (w_3,w_4) \in R\ Fr\ \hat{}\ n \wedge$
$(w_2,w_4) \in R\ Fr\ \hat{}\ k)\}$

lemma *R-n-closed*:
  assumes $g \in G\ Fr\ (g,h) \in R\ Fr\ \hat{}\ n$
  shows $h \in G\ Fr$
using *assms* **by** (*induct n*) *auto*

theorem *char* ($C'\ m\ n\ j\ k$ :: *'g KripkeClass*) $(\Diamond^m\Box^n\ (V\ p) \longrightarrow_M \Box^j\Diamond^k\ (V\ p))$
proof (*rule*, *rule*, *rule*)
  fix *Fr* :: *'g KripkeFrame*
  fix v
  assume *asm*: $Fr \in C'\ m\ n\ j\ k$
  show $(Fr,\ v) \models \Diamond^m\Box^n\ (V\ p) \longrightarrow_M \Box^j\Diamond^k\ (V\ p)$
  proof (*rule*, *rule*)
    fix w_1
    assume w_1: $w_1 \in G'\ (Fr,\ v)\ \langle(Fr,\ v),w_1\rangle \models \Diamond^m\Box^n\ (V\ p)$
    then obtain w_3 **where** w_3: $w_3 \in G\ Fr\ (w_1,w_3) \in R\ Fr\ \hat{}\ m\ \langle(Fr,v),w_3\rangle \models$
$\Box^n\ (V\ p)$ **by** *auto*
    from *this*(*1*,*3*) **have** p: $\forall w_4 \in G\ Fr.\ (w_3,w_4) \in R\ Fr\ \hat{}\ n \longrightarrow \langle(Fr,v),w_4\rangle \models$
$(V\ p)$ **by** *simp*
    from w_1(*1*) **show** $\langle(Fr,\ v),w_1\rangle \models \Box^j\Diamond^k\ (V\ p)$
    proof
      fix w_2
      assume w_2: $w_2 \in G'\ (Fr,\ v)\ (w_1,\ w_2) \in R'\ (Fr,\ v)\ \hat{}\ j$
      from *asm* w_1(*1*) w_3 *this* **obtain** w_4 **where** w_4: $(w_3,w_4) \in R\ Fr\ \hat{}\ n\ (w_2,w_4)$
$\in R\ Fr\ \hat{}\ k$ **by** *atomize-elim auto*
      from *this*(*1*) w_3(*1*) **have**[*simp*]: $w_4 \in G\ Fr$ **by** $-$(*rule R-n-closed*)
      from w_4(*1*) **have** $\langle(Fr,v),w_4\rangle \models (V\ p)$ **using** p[*THEN bspec*[**where** x=w_4]]
by *simp*
      with w_2(*1*) w_4(*2*) **show** $\langle(Fr,\ v),w_2\rangle \models \Diamond^k\ (V\ p)$ **by** *auto*
    qed
  qed
next
  fix *Fr* :: *'g KripkeFrame*
  assume *asm*: $\forall v.\ (Fr,\ v) \models \Diamond^m\Box^n\ (V\ p) \longrightarrow_M \Box^j\Diamond^k\ (V\ p)$

show $Fr \in C'\ m\ n\ j\ k$
proof (*simp*, *rule*, *rule*, *rule*, *rule*)
fix $w_1\ w_2\ w_3$
assume [*simp*]: $w_1 \in G\ Fr\ w_2 \in G\ Fr\ w_3 \in G\ Fr$
assume *123*: $(w_1,\ w_3) \in R\ Fr\ \hat{}\ m \wedge (w_1,\ w_2) \in R\ Fr\ \hat{}\ j$
let $?v = \lambda g\ p.\ (w_3,g) \in R\ Fr\ \hat{}\ n$
have $\langle(Fr,?v),w_1\rangle \models \Diamond^m\Box^n(V\ p)$ **using** *123* **by** *auto*
with *asm*[*THEN spec*[**where** $x= ?v$], *THEN bspec*[**where** $x=w_1$]] **have** $\langle(Fr,?v),w_1\rangle \models \Box^j\Diamond^k\ (V\ p)$ **by** *auto*
with *123* **obtain** w_4 **where** $(w_2,w_4) \in R\ Fr\ \hat{}\ k\ (w_3,w_4) \in R\ Fr\ \hat{}\ n$ **by** *auto*
thus $\exists\, w_4.\ (w_3,\ w_4) \in R\ Fr\ \hat{}\ n \wedge (w_2,\ w_4) \in R\ Fr\ \hat{}\ k$ **by** *auto*
qed
qed

3.4 Meaning of Some C-Properties

lemma[*simp*]: $r\ \hat{}\ 2 = r\ O\ r$ **by** (*simp add*:*Suc-1*[*symmetric*])

lemma $K4 = C\ 0\ 1\ 2\ 0$ **by** (*auto simp add*:*trans-def*)
lemma $T = C'\ 0\ 1\ 0\ 0$ **by** (*rule set-eqI*, *subst T*, *auto*)
lemma $\{Fr \in K.\ sym\ (R\ Fr)\} = C\ 1\ 1\ 0\ 0$ **by** (*simp add*:*sym-def*)

end