Embedding of the Theory of Abstract Objects in Isabelle/HOL

Daniel Kirchner

May 30, 2017

Abstract

This document constitutes a core contribution of the MSc project of Daniel Kirchner. The supervisor of this project is Christoph Benzmüller. The project idea results from an ongoing collaboration between Benzmüller and Zalta since 2015 and from the Computational Metaphysics lecture course held at FU Berlin in 2016.

Contents

1 Embedding

1.1 Primitives

typedecl i — possible worlds
typedecl j — states

consts dw :: i — actual world
consts dj :: j — actual state

typedecl ω — ordinary objects
typedecl σ — special urelements
datatype $\upsilon = \omega\upsilon\ \omega \mid \sigma\upsilon\ \sigma$ — urelements

1.2 Derived Types

typedef o = $UNIV::(j{\Rightarrow}i{\Rightarrow}bool)\ set$
morphisms $eval\mathrm{o}\ make\mathrm{o}$ **..** — truth values

type-synonym $\Pi_0 = \mathrm{o}$ — zero place relations
typedef $\Pi_1 = UNIV::(\upsilon{\Rightarrow}j{\Rightarrow}i{\Rightarrow}bool)\ set$
morphisms $eval\Pi_1\ make\Pi_1$ **..** — one place relations
typedef $\Pi_2 = UNIV::(\upsilon{\Rightarrow}\upsilon{\Rightarrow}j{\Rightarrow}i{\Rightarrow}bool)\ set$
morphisms $eval\Pi_2\ make\Pi_2$ **..** — two place relations
typedef $\Pi_3 = UNIV::(\upsilon{\Rightarrow}\upsilon{\Rightarrow}\upsilon{\Rightarrow}j{\Rightarrow}i{\Rightarrow}bool)\ set$
morphisms $eval\Pi_3\ make\Pi_3$ **..** — three place relations

type-synonym $\alpha = \Pi_1\ set$ — abstract objects

datatype $\nu = \omega\nu\ \omega \mid \alpha\nu\ \alpha$ — individuals

typedef $\kappa = UNIV::(\nu\ option)\ set$
morphisms $eval\kappa\ make\kappa$ **..** — individual terms

setup-lifting *type-definition-*o
setup-lifting *type-definition-*κ

setup-lifting *type-definition*-Π_1
setup-lifting *type-definition*-Π_2
setup-lifting *type-definition*-Π_3

1.3 Individual Terms and Definite Descriptions

Remark 1. *Individual terms can be definite descriptions which may not denote. Therefore the type for individual terms κ is defined as ν option. Individuals are represented by Some x for an individual x of type ν, whereas non-denoting individual terms are represented by None. Note that relation terms on the other hand always denote, so there is no need for a similar distinction between relation terms and relations.*

lift-definition $\nu\kappa$ :: $\nu \Rightarrow \kappa$ (-^{P} [*90*] *90*) **is** *Some* **.**
lift-definition *proper* :: $\kappa \Rightarrow$*bool* **is** *op*$\neq$ *None* **.**
lift-definition *rep* :: $\kappa \Rightarrow \nu$ **is** *the* **.**

Remark 2. *Individual terms can be explicitly marked to only range over logically proper objects (e.g. x^P). Their logical propriety and (in case they are logically proper) the represented individual can be extracted from the internal representation as ν option.*

lift-definition *that*::($\nu \Rightarrow$o)$\Rightarrow\kappa$ (**binder** $\boldsymbol{\iota}$ [*8*] *9*) **is**
λ φ . *if* ($\exists$! x . (φ x) *dj dw*)
then Some (*THE* x . (φ x) *dj dw*)
else None **.**

Remark 3. *Definite descriptions map conditions on individuals to individual terms. If no unique object satisfying the condition exists (and therefore the definite description is not logically proper), the individual term is set to None.*

1.4 Mapping from objects to urelements

consts $\alpha\sigma$ :: $\alpha \Rightarrow \sigma$
axiomatization where $\alpha\sigma$-*surj*: *surj* $\alpha\sigma$
definition $\nu\upsilon$ :: $\nu \Rightarrow \upsilon$ **where** $\nu\upsilon \equiv$ *case*-ν $\omega\upsilon$ ($\sigma\upsilon \circ \alpha\sigma$)

1.5 Exemplification of n-place relations.

lift-definition *exe0*::$\Pi_0 \Rightarrow$o ((|-|)) **is** *id* **.**
lift-definition *exe1*::$\Pi_1 \Rightarrow \kappa \Rightarrow$o ((|-,-|)) **is**
λ F x s w . (*proper* x) $\wedge$ F ($\nu\upsilon$ (*rep* x)) s w **.**
lift-definition *exe2*::$\Pi_2 \Rightarrow \kappa \Rightarrow \kappa \Rightarrow$o ((|-,-,-|)) **is**
λ F x y s w . (*proper* x) $\wedge$ (*proper* y) $\wedge$
F ($\nu\upsilon$ (*rep* x)) ($\nu\upsilon$ (*rep* y)) s w **.**
lift-definition *exe3*::$\Pi_3 \Rightarrow \kappa \Rightarrow \kappa \Rightarrow \kappa \Rightarrow$o ((|-,-,-,-|)) **is**
λ F x y z s w . (*proper* x) $\wedge$ (*proper* y) $\wedge$ (*proper* z) $\wedge$
F ($\nu\upsilon$ (*rep* x)) ($\nu\upsilon$ (*rep* y)) ($\nu\upsilon$ (*rep* z)) s w **.**

Remark 4. *An exemplification formula can only be true if all individual terms are logically proper. Furthermore exemplification depends on the urelement corresponding to the individual, not the individual itself.*

1.6 Encoding

lift-definition *enc* :: $\kappa \Rightarrow \Pi_1 \Rightarrow$o ({|-,-|}) **is**
λ x F s w . (*proper* x) $\wedge$ *case*-ν (λ ω . *False*) (λ α . $F \in \alpha$) (*rep* x) **.**

Remark 5. *An encoding formula can only be true if the individual term is logically proper. Furthermore ordinary objects never encode, whereas abstract objects encode a property if and only if the property is contained in it.*

1.7 Connectives and Quantifiers

consts $\mathit{I\text{-}NOT} :: j \Rightarrow (i \Rightarrow bool) \Rightarrow i \Rightarrow bool$
consts $\mathit{I\text{-}IMPL} :: j \Rightarrow (i \Rightarrow bool) \Rightarrow (i \Rightarrow bool) \Rightarrow (i \Rightarrow bool)$

lift-definition $not :: \mathrm{o} \Rightarrow \mathrm{o}$ ($\neg$- [*54*] *70*) **is**
$\lambda\ p\ s\ w\ .\ s = dj \wedge \neg p\ dj\ w \vee s \neq dj \wedge (\mathit{I\text{-}NOT}\ s\ (p\ s)\ w)$ **.**
lift-definition $impl :: \mathrm{o} \Rightarrow \mathrm{o} \Rightarrow \mathrm{o}$ (**infixl** $\rightarrow$ *51*) **is**
$\lambda\ p\ q\ s\ w\ .\ s = dj \wedge (p\ dj\ w \longrightarrow q\ dj\ w) \vee s \neq dj \wedge (\mathit{I\text{-}IMPL}\ s\ (p\ s)\ (q\ s)\ w)$ **.**
lift-definition $forall_\nu :: (\nu \Rightarrow \mathrm{o}) \Rightarrow \mathrm{o}$ (**binder** $\forall_\nu$ [*8*] *9*) **is**
$\lambda\ \varphi\ s\ w\ .\ \forall\ x :: \nu\ .\ (\varphi\ x)\ s\ w$ **.**
lift-definition $forall_0 :: (\Pi_0 \Rightarrow \mathrm{o}) \Rightarrow \mathrm{o}$ (**binder** $\forall_0$ [*8*] *9*) **is**
$\lambda\ \varphi\ s\ w\ .\ \forall\ x :: \Pi_0\ .\ (\varphi\ x)\ s\ w$ **.**
lift-definition $forall_1 :: (\Pi_1 \Rightarrow \mathrm{o}) \Rightarrow \mathrm{o}$ (**binder** $\forall_1$ [*8*] *9*) **is**
$\lambda\ \varphi\ s\ w\ .\ \forall\ x :: \Pi_1\ .\ (\varphi\ x)\ s\ w$ **.**
lift-definition $forall_2 :: (\Pi_2 \Rightarrow \mathrm{o}) \Rightarrow \mathrm{o}$ (**binder** $\forall_2$ [*8*] *9*) **is**
$\lambda\ \varphi\ s\ w\ .\ \forall\ x :: \Pi_2\ .\ (\varphi\ x)\ s\ w$ **.**
lift-definition $forall_3 :: (\Pi_3 \Rightarrow \mathrm{o}) \Rightarrow \mathrm{o}$ (**binder** $\forall_3$ [*8*] *9*) **is**
$\lambda\ \varphi\ s\ w\ .\ \forall\ x :: \Pi_3\ .\ (\varphi\ x)\ s\ w$ **.**
lift-definition $forall_\mathrm{o} :: (\mathrm{o} \Rightarrow \mathrm{o}) \Rightarrow \mathrm{o}$ (**binder** $\forall_\mathrm{o}$ [*8*] *9*) **is**
$\lambda\ \varphi\ s\ w\ .\ \forall\ x :: \mathrm{o}\ .\ (\varphi\ x)\ s\ w$ **.**
lift-definition $box :: \mathrm{o} \Rightarrow \mathrm{o}$ ($\Box$- [*62*] *63*) **is**
$\lambda\ p\ s\ w\ .\ \forall\ v\ .\ p\ s\ v$ **.**
lift-definition $actual :: \mathrm{o} \Rightarrow \mathrm{o}$ ($\boldsymbol{\mathcal{A}}$- [*64*] *65*) **is**
$\lambda\ p\ s\ w\ .\ p\ s\ dw$ **.**

Remark 6. *The connectives behave classically if evaluated for the actual state dj, whereas their behavior is governed by uninterpreted constants for any other state.*

1.8 Lambda Expressions

Remark 7. *Lambda expressions have to convert maps from individuals to propositions to relations that are represented by maps from urelements to truth values.*

lift-definition $lambdabinder0 :: \mathrm{o} \Rightarrow \Pi_0$ ($\boldsymbol{\lambda}^0$) **is** *id* **.**
lift-definition $lambdabinder1 :: (\nu \Rightarrow \mathrm{o}) \Rightarrow \Pi_1$ (**binder** $\boldsymbol{\lambda}$ [*8*] *9*) **is**
$\lambda\ \varphi\ u\ s\ w\ .\ \exists\ x\ .\ \nu\upsilon\ x = u \wedge \varphi\ x\ s\ w$ **.**
lift-definition $lambdabinder2 :: (\nu \Rightarrow \nu \Rightarrow \mathrm{o}) \Rightarrow \Pi_2$ ($\boldsymbol{\lambda}^2$) **is**
$\lambda\ \varphi\ u\ v\ s\ w\ .\ \exists\ x\ y\ .\ \nu\upsilon\ x = u \wedge \nu\upsilon\ y = v \wedge \varphi\ x\ y\ s\ w$ **.**
lift-definition $lambdabinder3 :: (\nu \Rightarrow \nu \Rightarrow \nu \Rightarrow \mathrm{o}) \Rightarrow \Pi_3$ ($\boldsymbol{\lambda}^3$) **is**
$\lambda\ \varphi\ u\ v\ r\ s\ w\ .\ \exists\ x\ y\ z\ .\ \nu\upsilon\ x = u \wedge \nu\upsilon\ y = v \wedge \nu\upsilon\ z = r \wedge \varphi\ x\ y\ z\ s\ w$ **.**

1.9 Proper Maps

Remark 8. *The embedding introduces the notion of* proper *maps from individual terms to propositions.*
Such a map is proper if and only if for all proper individual terms its truth evaluation in the actual state only depends on the urelements corresponding to the individuals the terms denote.
Proper maps are exactly those maps that - when used in a lambda-expression - unconditionally allow beta-reduction.

lift-definition $IsProperInX :: (\kappa \Rightarrow \mathrm{o}) \Rightarrow bool$ **is**
$\lambda\ \varphi\ .\ \forall\ x\ v\ .\ (\exists\ a\ .\ \nu\upsilon\ a = \nu\upsilon\ x \wedge (\varphi\ (a^P)\ dj\ v)) = (\varphi\ (x^P)\ dj\ v)$ **.**
lift-definition $IsProperInXY :: (\kappa \Rightarrow \kappa \Rightarrow \mathrm{o}) \Rightarrow bool$ **is**
$\lambda\ \varphi\ .\ \forall\ x\ y\ v\ .\ (\exists\ a\ b\ .\ \nu\upsilon\ a = \nu\upsilon\ x \wedge \nu\upsilon\ b = \nu\upsilon\ y$
$\wedge (\varphi\ (a^P)\ (b^P)\ dj\ v)) = (\varphi\ (x^P)\ (y^P)\ dj\ v)$ **.**
lift-definition $IsProperInXYZ :: (\kappa \Rightarrow \kappa \Rightarrow \kappa \Rightarrow \mathrm{o}) \Rightarrow bool$ **is**
$\lambda\ \varphi\ .\ \forall\ x\ y\ z\ v\ .\ (\exists\ a\ b\ c\ .\ \nu\upsilon\ a = \nu\upsilon\ x \wedge \nu\upsilon\ b = \nu\upsilon\ y \wedge \nu\upsilon\ c = \nu\upsilon\ z$
$\wedge (\varphi\ (a^P)\ (b^P)\ (c^P)\ dj\ v)) = (\varphi\ (x^P)\ (y^P)\ (z^P)\ dj\ v)$ **.**

1.10 Validity

lift-definition *valid-in* :: $i \Rightarrow o \Rightarrow bool$ (**infixl** $\models$ *5*) **is**
$\lambda\ v\ \varphi\ .\ \varphi\ dj\ v$ **.**

Remark 9. *A formula is considered semantically valid for a possible world, if it evaluates to True for the actual state dj and the given possible world.*

1.11 Concreteness

consts *ConcreteInWorld* :: $\omega \Rightarrow i \Rightarrow bool$

abbreviation (*input*) *OrdinaryObjectsPossiblyConcrete* **where**
OrdinaryObjectsPossiblyConcrete $\equiv \forall\ x\ .\ \exists\ v\ .$ *ConcreteInWorld x v*
abbreviation (*input*) *PossiblyContingentObjectExists* **where**
PossiblyContingentObjectExists $\equiv \exists\ x\ v\ .$ *ConcreteInWorld x v*
$\wedge\ (\exists\ w\ .\ \neg$ *ConcreteInWorld x w*)
abbreviation (*input*) *PossiblyNoContingentObjectExists* **where**
PossiblyNoContingentObjectExists $\equiv \exists\ w\ .\ \forall\ x\ .$ *ConcreteInWorld x w*
$\longrightarrow (\forall\ v\ .$ *ConcreteInWorld x v*)
axiomatization where
OrdinaryObjectsPossiblyConcreteAxiom:
OrdinaryObjectsPossiblyConcrete
and *PossiblyContingentObjectExistsAxiom*:
PossiblyContingentObjectExists
and *PossiblyNoContingentObjectExistsAxiom*:
PossiblyNoContingentObjectExists

Remark 10. *Care has to be taken that the defined notion of concreteness coincides with the meta-logical distinction between abstract objects and ordinary objects. Furthermore the axioms about concreteness have to be satisfied. This is achieved by introducing an uninterpreted constant ConcreteInWorld that determines whether an ordinary object is concrete in a given possible world. This constant is axiomatized, such that all ordinary objects are possibly concrete, contingent objects possibly exist and possibly no contingent objects exist.*

lift-definition *Concrete*::Π_1 ($E!$) **is**
$\lambda\ u\ s\ w\ .$ *case u of* $\omega\upsilon\ x \Rightarrow$ *ConcreteInWorld x w* | - $\Rightarrow$ *False* **.**

Remark 11. *Concreteness of ordinary objects is now defined using this axiomatized uninterpreted constant. Abstract objects on the other hand are never concrete.*

1.12 Collection of Meta-Definitions

named-theorems *meta-defs*

declare *not-def*[*meta-defs*] *impl-def*[*meta-defs*] *forall*$_\nu$*-def*[*meta-defs*]
forall$_0$*-def*[*meta-defs*] *forall*$_1$*-def*[*meta-defs*]
forall$_2$*-def*[*meta-defs*] *forall*$_3$*-def*[*meta-defs*] *forall*$_o$*-def*[*meta-defs*]
box-def[*meta-defs*] *actual-def*[*meta-defs*] *that-def*[*meta-defs*]
lambdabinder0-def[*meta-defs*] *lambdabinder1-def*[*meta-defs*]
lambdabinder2-def[*meta-defs*] *lambdabinder3-def*[*meta-defs*]
exe0-def[*meta-defs*] *exe1-def*[*meta-defs*] *exe2-def*[*meta-defs*]
exe3-def[*meta-defs*] *enc-def*[*meta-defs*] *inv-def*[*meta-defs*]
that-def[*meta-defs*] *valid-in-def*[*meta-defs*] *Concrete-def*[*meta-defs*]

declare [[*smt-solver = cvc4*]]
declare [[*simp-depth-limit = 10*]]
declare [[*unify-search-bound = 40*]]

1.13 Auxiliary Lemmata

named-theorems *meta-aux*

declare *make*κ*-inverse*[*meta-aux*] *eval*κ*-inverse*[*meta-aux*]
*make*o*-inverse*[*meta-aux*] *eval*o*-inverse*[*meta-aux*]
makeΠ_1*-inverse*[*meta-aux*] *eval*Π_1*-inverse*[*meta-aux*]
makeΠ_2*-inverse*[*meta-aux*] *eval*Π_2*-inverse*[*meta-aux*]
makeΠ_3*-inverse*[*meta-aux*] *eval*Π_3*-inverse*[*meta-aux*]
lemma $\nu\upsilon$*-*$\omega\nu$*-is-*$\omega\upsilon$[*meta-aux*]: $\nu\upsilon\ (\omega\nu\ x) = \omega\upsilon\ x$ **by** (*simp add*: $\nu\upsilon$*-def*)
lemma *rep-proper-id*[*meta-aux*]: *rep* $(x^P) = x$
by (*simp add*: *meta-aux* $\nu\kappa$*-def rep-def*)
lemma $\nu\kappa$*-proper*[*meta-aux*]: *proper* (x^P)
by (*simp add*: *meta-aux* $\nu\kappa$*-def proper-def*)
lemma *no-*$\alpha\omega$[*meta-aux*]: $\neg(\nu\upsilon\ (\alpha\nu\ x) = \omega\upsilon\ y)$ **by** (*simp add*: $\nu\upsilon$*-def*)
lemma *no-*$\sigma\omega$[*meta-aux*]: $\neg(\sigma\upsilon\ x = \omega\upsilon\ y)$ **by** *blast*
lemma $\nu\upsilon$*-surj*[*meta-aux*]: *surj* $\nu\upsilon$
using $\alpha\sigma$*-surj* **unfolding** $\nu\upsilon$*-def surj-def*
by (*metis* ν*.simps*(5) ν*.simps*(6) υ*.exhaust comp-apply*)
lemma *lambda*Π_1*-aux*[*meta-aux*]:
makeΠ_1 $(\lambda u\ s\ w.\ \exists x.\ \nu\upsilon\ x = u \wedge eval\Pi_1\ F\ (\nu\upsilon\ x)\ s\ w) = F$
proof −
have $\bigwedge u\ s\ w\ \varphi\ .\ (\exists\ x\ .\ \nu\upsilon\ x = u \wedge \varphi\ (\nu\upsilon\ x)\ (s{::}j)\ (w{::}i)) \longleftrightarrow \varphi\ u\ s\ w$
using $\nu\upsilon$*-surj* **unfolding** *surj-def* **by** *metis*
thus *?thesis* **apply** *transfer* **by** *simp*
qed
lemma *lambda*Π_2*-aux*[*meta-aux*]:
makeΠ_2 $(\lambda u\ v\ s\ w.\ \exists x\ .\ \nu\upsilon\ x = u \wedge (\exists\ y\ .\ \nu\upsilon\ y = v \wedge eval\Pi_2\ F\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ s\ w)) = F$
proof −
have $\bigwedge u\ v\ (s\ {::}j)\ (w{::}i)\ \varphi\ .$
$(\exists\ x\ .\ \nu\upsilon\ x = u \wedge (\exists\ y\ .\ \nu\upsilon\ y = v \wedge \varphi\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ s\ w))$
$\longleftrightarrow \varphi\ u\ v\ s\ w$
using $\nu\upsilon$*-surj* **unfolding** *surj-def* **by** *metis*
thus *?thesis* **apply** *transfer* **by** *simp*
qed
lemma *lambda*Π_3*-aux*[*meta-aux*]:
makeΠ_3 $(\lambda u\ v\ r\ s\ w.\ \exists x.\ \nu\upsilon\ x = u \wedge (\exists y.\ \nu\upsilon\ y = v \wedge$
$(\exists z.\ \nu\upsilon\ z = r \wedge eval\Pi_3\ F\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ (\nu\upsilon\ z)\ s\ w))) = F$
proof −
have $\bigwedge u\ v\ r\ (s{::}j)\ (w{::}i)\ \varphi\ .\ \exists x.\ \nu\upsilon\ x = u \wedge (\exists y.\ \nu\upsilon\ y = v$
$\wedge (\exists z.\ \nu\upsilon\ z = r \wedge \varphi\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ (\nu\upsilon\ z)\ s\ w)) = \varphi\ u\ v\ r\ s\ w$
using $\nu\upsilon$*-surj* **unfolding** *surj-def* **by** *metis*
thus *?thesis* **apply** *transfer* **apply** (*rule ext*)+ **by** *metis*
qed

2 Semantics

2.1 Definition

locale *Semantics*
begin
named-theorems *semantics*

2.1.1 Semantical Domains

type-synonym $R_\kappa = \nu$
type-synonym $R_0 = j \Rightarrow i \Rightarrow bool$
type-synonym $R_1 = \upsilon \Rightarrow R_0$
type-synonym $R_2 = \upsilon \Rightarrow \upsilon \Rightarrow R_0$
type-synonym $R_3 = \upsilon \Rightarrow \upsilon \Rightarrow \upsilon \Rightarrow R_0$
type-synonym $W = i$

2.1.2 Denotation Functions

lift-definition d_κ :: $\kappa \Rightarrow R_\kappa$ *option* **is** *id* **.**
lift-definition d_0 :: $\Pi_0 \Rightarrow R_0$ *option* **is** *Some* **.**
lift-definition d_1 :: $\Pi_1 \Rightarrow R_1$ *option* **is** *Some* **.**
lift-definition d_2 :: $\Pi_2 \Rightarrow R_2$ *option* **is** *Some* **.**
lift-definition d_3 :: $\Pi_3 \Rightarrow R_3$ *option* **is** *Some* **.**

2.1.3 Actual World

definition w_0 **where** $w_0 \equiv dw$

2.1.4 Exemplification Extensions

definition *ex0* :: $R_0 \Rightarrow W \Rightarrow bool$
  where *ex0* $\equiv \lambda\ F\ .\ F\ dj$
definition *ex1* :: $R_1 \Rightarrow W \Rightarrow (R_\kappa\ set)$
  where *ex1* $\equiv \lambda\ F\ w\ .\ \{\ x\ .\ F\ (\nu\upsilon\ x)\ dj\ w\ \}$
definition *ex2* :: $R_2 \Rightarrow W \Rightarrow ((R_\kappa \times R_\kappa)\ set)$
  where *ex2* $\equiv \lambda\ F\ w\ .\ \{\ (x,y)\ .\ F\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ dj\ w\ \}$
definition *ex3* :: $R_3 \Rightarrow W \Rightarrow ((R_\kappa \times R_\kappa \times R_\kappa)\ set)$
  where *ex3* $\equiv \lambda\ F\ w\ .\ \{\ (x,y,z)\ .\ F\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ (\nu\upsilon\ z)\ dj\ w\ \}$

2.1.5 Encoding Extensions

definition *en* :: $R_1 \Rightarrow (R_\kappa\ set)$
  where *en* $\equiv \lambda\ F\ .\ \{\ x\ .\ case\ x\ of\ \alpha\nu\ y \Rightarrow make\Pi_1\ (\lambda\ x\ .\ F\ x) \in y$
                                 $|\ \text{-} \Rightarrow False\ \}$

2.1.6 Collection of Semantical Definitions

named-theorems *semantics-defs*
declare d_0-*def*[*semantics-defs*] d_1-*def*[*semantics-defs*]
    d_2-*def*[*semantics-defs*] d_3-*def*[*semantics-defs*]
    ex0-def[*semantics-defs*] *ex1-def*[*semantics-defs*]
    ex2-def[*semantics-defs*] *ex3-def*[*semantics-defs*]
    en-def[*semantics-defs*] d_κ-*def*[*semantics-defs*]
    w_0-*def*[*semantics-defs*]

2.1.7 Truth Conditions of Exemplification Formulas

lemma *T1-1*[*semantics*]:
  $(w \models (\!|F,x|\!)) = (\exists\ r\ o_1\ .\ Some\ r = d_1\ F \wedge Some\ o_1 = d_\kappa\ x \wedge o_1 \in ex1\ r\ w)$
  unfolding *semantics-defs*
  apply (*simp add*: *meta-defs meta-aux rep-def proper-def*)
  by (*metis option.discI option.exhaust option.sel*)

lemma *T1-2*[*semantics*]:
  $(w \models (\!|F,x,y|\!)) = (\exists\ r\ o_1\ o_2\ .\ Some\ r = d_2\ F \wedge Some\ o_1 = d_\kappa\ x$
              $\wedge\ Some\ o_2 = d_\kappa\ y \wedge (o_1,\ o_2) \in ex2\ r\ w)$
  unfolding *semantics-defs*
  apply (*simp add*: *meta-defs meta-aux rep-def proper-def*)
  by (*metis option.discI option.exhaust option.sel*)

lemma *T1-3*[*semantics*]:
  $(w \models (\!|F,x,y,z|\!)) = (\exists\ r\ o_1\ o_2\ o_3\ .\ Some\ r = d_3\ F \wedge Some\ o_1 = d_\kappa\ x$
              $\wedge\ Some\ o_2 = d_\kappa\ y \wedge Some\ o_3 = d_\kappa\ z$
              $\wedge\ (o_1,\ o_2,\ o_3) \in ex3\ r\ w)$
  unfolding *semantics-defs*
  apply (*simp add*: *meta-defs meta-aux rep-def proper-def*)
  by (*metis option.discI option.exhaust option.sel*)

lemma *T3*[*semantics*]:
$(w \models (\!|F|\!)) = (\exists\ r\ .\ Some\ r = d_0\ F \wedge ex0\ r\ w)$
unfolding *semantics-defs*
by (*simp add*: *meta-defs meta-aux*)

2.1.8 Truth Conditions of Encoding Formulas

lemma *T2*[*semantics*]:
$(w \models \{\!|x,F|\!\}) = (\exists\ r\ o_1\ .\ Some\ r = d_1\ F \wedge Some\ o_1 = d_\kappa\ x \wedge o_1 \in en\ r)$
unfolding *semantics-defs*
apply (*simp add*: *meta-defs meta-aux rep-def proper-def split*: *ν.split*)
by (*metis ν.exhaust ν.inject(2) ν.simps(4) νκ.rep-eq option.collapse option.discI rep.rep-eq rep-proper-id*)

2.1.9 Truth Conditions of Complex Formulas

lemma *T4*[*semantics*]: $(w \models \neg\psi) = (\neg(w \models \psi))$
by (*simp add*: *meta-defs meta-aux*)

lemma *T5*[*semantics*]: $(w \models \psi \rightarrow \chi) = (\neg(w \models \psi) \vee (w \models \chi))$
by (*simp add*: *meta-defs meta-aux*)

lemma *T6*[*semantics*]: $(w \models \Box\psi) = (\forall\ v\ .\ (v \models \psi))$
by (*simp add*: *meta-defs meta-aux*)

lemma *T7*[*semantics*]: $(w \models \boldsymbol{\mathcal{A}}\psi) = (dw \models \psi)$
by (*simp add*: *meta-defs meta-aux*)

lemma *T8-ν*[*semantics*]: $(w \models \forall_\nu\ x.\ \psi\ x) = (\forall\ x\ .\ (w \models \psi\ x))$
by (*simp add*: *meta-defs meta-aux*)

lemma *T8-0*[*semantics*]: $(w \models \forall_0\ x.\ \psi\ x) = (\forall\ x\ .\ (w \models \psi\ x))$
by (*simp add*: *meta-defs meta-aux*)

lemma *T8-1*[*semantics*]: $(w \models \forall_1\ x.\ \psi\ x) = (\forall\ x\ .\ (w \models \psi\ x))$
by (*simp add*: *meta-defs meta-aux*)

lemma *T8-2*[*semantics*]: $(w \models \forall_2\ x.\ \psi\ x) = (\forall\ x\ .\ (w \models \psi\ x))$
by (*simp add*: *meta-defs meta-aux*)

lemma *T8-3*[*semantics*]: $(w \models \forall_3\ x.\ \psi\ x) = (\forall\ x\ .\ (w \models \psi\ x))$
by (*simp add*: *meta-defs meta-aux*)

lemma *T8-*o[*semantics*]: $(w \models \forall_{\mathrm{o}}\ x.\ \psi\ x) = (\forall\ x\ .\ (w \models \psi\ x))$
by (*simp add*: *meta-defs meta-aux*)

2.1.10 Denotations of Descriptions

lemma *D3*[*semantics*]:
$d_\kappa\ (\iota x\ .\ \psi\ x) = (if\ (\exists x\ .\ (w_0 \models \psi\ x) \wedge (\forall\ y\ .\ (w_0 \models \psi\ y) \longrightarrow y = x))$
$then\ (Some\ (THE\ x\ .\ (w_0 \models \psi\ x)))\ else\ None)$
unfolding *semantics-defs*
by (*auto simp*: *meta-defs meta-aux*)

2.1.11 Denotations of Lambda Expressions

lemma *D4-1*[*semantics*]: $d_1\ (\boldsymbol{\lambda}\ x\ .\ (\!|F,\ x^P|\!)) = d_1\ F$
by (*simp add*: *meta-defs meta-aux*)

lemma *D4-2*[*semantics*]: $d_2\ (\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ (\!|F,\ x^P,\ y^P|\!))) = d_2\ F$
by (*simp add*: *meta-defs meta-aux*)

lemma *D4-3*[*semantics*]: d_3 ($\boldsymbol{\lambda}^3$ (λ x y z . ⦇F, x^P, y^P, z^P⦈)) = d_3 F
  by (*simp add*: *meta-defs meta-aux*)

lemma *D5-1*[*semantics*]:
  assumes *IsProperInX* φ
  shows $\bigwedge$ w o_1 r . *Some* r = d_1 ($\boldsymbol{\lambda}$ x . (φ (x^P))) $\wedge$ *Some* o_1 = d_κ x
        $\longrightarrow$ (o_1 $\in$ *ex1* r w) = (w $\models$ φ x)
  using *assms* **unfolding** *IsProperInX-def semantics-defs*
  by (*auto simp*: *meta-defs meta-aux rep-def proper-def* $\nu\kappa$*.abs-eq*)

lemma *D5-2*[*semantics*]:
  assumes *IsProperInXY* φ
  shows $\bigwedge$ w o_1 o_2 r . *Some* r = d_2 ($\boldsymbol{\lambda}^2$ (λ x y . φ (x^P) (y^P)))
        $\wedge$ *Some* o_1 = d_κ x $\wedge$ *Some* o_2 = d_κ y
        $\longrightarrow$ ((o_1,o_2) $\in$ *ex2* r w) = (w $\models$ φ x y)
  using *assms* **unfolding** *IsProperInXY-def semantics-defs*
  by (*auto simp*: *meta-defs meta-aux rep-def proper-def* $\nu\kappa$*.abs-eq*)

lemma *D5-3*[*semantics*]:
  assumes *IsProperInXYZ* φ
  shows $\bigwedge$ w o_1 o_2 o_3 r . *Some* r = d_3 ($\boldsymbol{\lambda}^3$ (λ x y z . φ (x^P) (y^P) (z^P)))
        $\wedge$ *Some* o_1 = d_κ x $\wedge$ *Some* o_2 = d_κ y $\wedge$ *Some* o_3 = d_κ z
        $\longrightarrow$ ((o_1,o_2,o_3) $\in$ *ex3* r w) = (w $\models$ φ x y z)
  using *assms* **unfolding** *IsProperInXYZ-def semantics-defs*
  by (*auto simp*: *meta-defs meta-aux rep-def proper-def* $\nu\kappa$*.abs-eq*)

lemma *D6*[*semantics*]: ($\bigwedge$ w r . *Some* r = d_0 ($\boldsymbol{\lambda}^0$ φ) $\longrightarrow$ *ex0* r w = (w $\models$ φ))
  by (*auto simp*: *meta-defs meta-aux semantics-defs*)

2.1.12 Auxiliary Lemmas

lemma *propex*$_0$: $\exists$ r . *Some* r = d_0 F
  unfolding d_0*-def* **by** *simp*
lemma *propex*$_1$: $\exists$ r . *Some* r = d_1 F
  unfolding d_1*-def* **by** *simp*
lemma *propex*$_2$: $\exists$ r . *Some* r = d_2 F
  unfolding d_2*-def* **by** *simp*
lemma *propex*$_3$: $\exists$ r . *Some* r = d_3 F
  unfolding d_3*-def* **by** *simp*
lemma d_κ*-proper*: d_κ (u^P) = *Some* u
  unfolding d_κ*-def* **by** (*simp add*: $\nu\kappa$*-def meta-aux*)
lemma *ConcretenessSemantics1*:
  Some r = d_1 *E*! $\Longrightarrow$ ($\exists$ w . $\omega\nu$ x $\in$ *ex1* r w)
  unfolding *semantics-defs* **apply** *transfer*
  by (*simp add*: *OrdinaryObjectsPossiblyConcreteAxiom* $\nu\upsilon$-$\omega\nu$-*is*-$\omega\upsilon$)
lemma *ConcretenessSemantics2*:
  Some r = d_1 *E*! $\Longrightarrow$ (x $\in$ *ex1* r w $\longrightarrow$ ($\exists$ y. x = $\omega\nu$ y))
  unfolding *semantics-defs* **apply** *transfer* **apply** *simp*
  by (*metis* ν*.exhaust* υ*.exhaust* υ*.simps*(*6*) *no*-$\alpha\omega$)
lemma d_0*-inject*: $\bigwedge x$ y. d_0 x = d_0 y $\Longrightarrow$ x = y
  unfolding d_0*-def* **by** (*simp add*: *eval*o*-inject*)
lemma d_1*-inject*: $\bigwedge x$ y. d_1 x = d_1 y $\Longrightarrow$ x = y
  unfolding d_1*-def* **by** (*simp add*: *eval*Π_1*-inject*)
lemma d_2*-inject*: $\bigwedge x$ y. d_2 x = d_2 y $\Longrightarrow$ x = y
  unfolding d_2*-def* **by** (*simp add*: *eval*Π_2*-inject*)
lemma d_3*-inject*: $\bigwedge x$ y. d_3 x = d_3 y $\Longrightarrow$ x = y
  unfolding d_3*-def* **by** (*simp add*: *eval*Π_3*-inject*)
lemma d_κ*-inject*: $\bigwedge x$ y o_1. *Some* o_1 = d_κ x $\wedge$ *Some* o_1 = d_κ y $\Longrightarrow$ x = y
proof −
  fix x :: κ **and** y :: κ **and** o_1 :: ν
  assume *Some* o_1 = d_κ x $\wedge$ *Some* o_1 = d_κ y
  thus x = y **apply** *transfer* **by** *auto*
qed

end

2.2 Introduction Rules for Proper Maps

Remark 12. *Every map whose argument only occurs in exemplification expressions is proper.*

named-theorems *IsProper-intros*

```
lemma IsProperInX-intro[IsProper-intros]:
  IsProperInX (λ x . χ
    (* one place *) (λ F . ⦇F,x⦈)
    (* two place *) (λ F . ⦇F,x,x⦈) (λ F a . ⦇F,x,a⦈) (λ F a . ⦇F,a,x⦈)
    (* three place three x *) (λ F . ⦇F,x,x,x⦈)
    (* three place two x *) (λ F a . ⦇F,x,x,a⦈) (λ F a . ⦇F,x,a,x⦈)
                            (λ F a . ⦇F,a,x,x⦈)
    (* three place one x *) (λ F a b. ⦇F,x,a,b⦈) (λ F a b. ⦇F,a,x,b⦈)
                            (λ F a b . ⦇F,a,b,x⦈))
  unfolding IsProperInX-def
  by (auto simp: meta-defs meta-aux)

lemma IsProperInXY-intro[IsProper-intros]:
  IsProperInXY (λ x y . χ
    (* only x *)
      (* one place *) (λ F . ⦇F,x⦈)
      (* two place *) (λ F . ⦇F,x,x⦈) (λ F a . ⦇F,x,a⦈) (λ F a . ⦇F,a,x⦈)
      (* three place three x *) (λ F . ⦇F,x,x,x⦈)
      (* three place two x *) (λ F a . ⦇F,x,x,a⦈) (λ F a . ⦇F,x,a,x⦈)
                              (λ F a . ⦇F,a,x,x⦈)
      (* three place one x *) (λ F a b. ⦇F,x,a,b⦈) (λ F a b. ⦇F,a,x,b⦈)
                              (λ F a b . ⦇F,a,b,x⦈)
    (* only y *)
      (* one place *) (λ F . ⦇F,y⦈)
      (* two place *) (λ F . ⦇F,y,y⦈) (λ F a . ⦇F,y,a⦈) (λ F a . ⦇F,a,y⦈)
      (* three place three y *) (λ F . ⦇F,y,y,y⦈)
      (* three place two y *) (λ F a . ⦇F,y,y,a⦈) (λ F a . ⦇F,y,a,y⦈)
                              (λ F a . ⦇F,a,y,y⦈)
      (* three place one y *) (λ F a b. ⦇F,y,a,b⦈) (λ F a b. ⦇F,a,y,b⦈)
                              (λ F a b . ⦇F,a,b,y⦈)
    (* x and y *)
      (* two place *) (λ F . ⦇F,x,y⦈) (λ F . ⦇F,y,x⦈)
      (* three place (x,y) *) (λ F a . ⦇F,x,y,a⦈) (λ F a . ⦇F,x,a,y⦈)
                              (λ F a . ⦇F,a,x,y⦈)
      (* three place (y,x) *) (λ F a . ⦇F,y,x,a⦈) (λ F a . ⦇F,y,a,x⦈)
                              (λ F a . ⦇F,a,y,x⦈)
      (* three place (x,x,y) *) (λ F . ⦇F,x,x,y⦈) (λ F . ⦇F,x,y,x⦈)
                                (λ F . ⦇F,y,x,x⦈)
      (* three place (x,y,y) *) (λ F . ⦇F,x,y,y⦈) (λ F . ⦇F,y,x,y⦈)
                                (λ F . ⦇F,y,y,x⦈)
      (* three place (x,x,x) *) (λ F . ⦇F,x,x,x⦈)
      (* three place (y,y,y) *) (λ F . ⦇F,y,y,y⦈))
  unfolding IsProperInXY-def by (auto simp: meta-defs meta-aux)

lemma IsProperInXYZ-intro[IsProper-intros]:
  IsProperInXYZ (λ x y z . χ
    (* only x *)
      (* one place *) (λ F . ⦇F,x⦈)
      (* two place *) (λ F . ⦇F,x,x⦈) (λ F a . ⦇F,x,a⦈) (λ F a . ⦇F,a,x⦈)
      (* three place three x *) (λ F . ⦇F,x,x,x⦈)
      (* three place two x *) (λ F a . ⦇F,x,x,a⦈) (λ F a . ⦇F,x,a,x⦈)
                              (λ F a . ⦇F,a,x,x⦈)
      (* three place one x *) (λ F a b. ⦇F,x,a,b⦈) (λ F a b. ⦇F,a,x,b⦈)
                              (λ F a b . ⦇F,a,b,x⦈)
    (* only y *)
```

```
    (* one place *) (λ F . (|F,y|))
    (* two place *) (λ F . (|F,y,y|)) (λ F a . (|F,y,a|)) (λ F a . (|F,a,y|))
    (* three place three y *) (λ F . (|F,y,y,y|))
    (* three place two y *) (λ F a . (|F,y,y,a|)) (λ F a . (|F,y,a,y|))
                            (λ F a . (|F,a,y,y|))
    (* three place one y *) (λ F a b. (|F,y,a,b|)) (λ F a b. (|F,a,y,b|))
                            (λ F a b . (|F,a,b,y|))
  (* only z *)
    (* one place *) (λ F . (|F,z|))
    (* two place *) (λ F . (|F,z,z|)) (λ F a . (|F,z,a|)) (λ F a . (|F,a,z|))
    (* three place three z *) (λ F . (|F,z,z,z|))
    (* three place two z *) (λ F a . (|F,z,z,a|)) (λ F a . (|F,z,a,z|))
                            (λ F a . (|F,a,z,z|))
    (* three place one z *) (λ F a b. (|F,z,a,b|)) (λ F a b. (|F,a,z,b|))
                            (λ F a b . (|F,a,b,z|))
  (* x and y *)
    (* two place *) (λ F . (|F,x,y|)) (λ F . (|F,y,x|))
    (* three place (x,y) *) (λ F a . (|F,x,y,a|)) (λ F a . (|F,x,a,y|))
                            (λ F a . (|F,a,x,y|))
    (* three place (y,x) *) (λ F a . (|F,y,x,a|)) (λ F a . (|F,y,a,x|))
                            (λ F a . (|F,a,y,x|))
    (* three place (x,x,y) *) (λ F . (|F,x,x,y|)) (λ F . (|F,x,y,x|))
                              (λ F . (|F,y,x,x|))
    (* three place (x,y,y) *) (λ F . (|F,x,y,y|)) (λ F . (|F,y,x,y|))
                              (λ F . (|F,y,y,x|))
    (* three place (x,x,x) *) (λ F . (|F,x,x,x|))
    (* three place (y,y,y) *) (λ F . (|F,y,y,y|))
  (* x and z *)
    (* two place *) (λ F . (|F,x,z|)) (λ F . (|F,z,x|))
    (* three place (x,z) *) (λ F a . (|F,x,z,a|)) (λ F a . (|F,x,a,z|))
                            (λ F a . (|F,a,x,z|))
    (* three place (z,x) *) (λ F a . (|F,z,x,a|)) (λ F a . (|F,z,a,x|))
                            (λ F a . (|F,a,z,x|))
    (* three place (x,x,z) *) (λ F . (|F,x,x,z|)) (λ F . (|F,x,z,x|))
                              (λ F . (|F,z,x,x|))
    (* three place (x,z,z) *) (λ F . (|F,x,z,z|)) (λ F . (|F,z,x,z|))
                              (λ F . (|F,z,z,x|))
    (* three place (x,x,x) *) (λ F . (|F,x,x,x|))
    (* three place (z,z,z) *) (λ F . (|F,z,z,z|))
  (* y and z *)
    (* two place *) (λ F . (|F,y,z|)) (λ F . (|F,z,y|))
    (* three place (y,z) *) (λ F a . (|F,y,z,a|)) (λ F a . (|F,y,a,z|))
                            (λ F a . (|F,a,y,z|))
    (* three place (z,y) *) (λ F a . (|F,z,y,a|)) (λ F a . (|F,z,a,y|))
                            (λ F a . (|F,a,z,y|))
    (* three place (y,y,z) *) (λ F . (|F,y,y,z|)) (λ F . (|F,y,z,y|))
                              (λ F . (|F,z,y,y|))
    (* three place (y,z,z) *) (λ F . (|F,y,z,z|)) (λ F . (|F,z,y,z|))
                              (λ F . (|F,z,z,y|))
    (* three place (y,y,y) *) (λ F . (|F,y,y,y|))
    (* three place (z,z,z) *) (λ F . (|F,z,z,z|))
  (* x y z *)
    (* three place (x,...) *) (λ F . (|F,x,y,z|)) (λ F . (|F,x,z,y|))
    (* three place (y,...) *) (λ F . (|F,y,x,z|)) (λ F . (|F,y,z,x|))
    (* three place (z,...) *) (λ F . (|F,z,x,y|)) (λ F . (|F,z,y,x|)))
 unfolding IsProperInXYZ-def
 by (auto simp: meta-defs meta-aux)

method show-proper = (fast intro: IsProper-intros)
```

2.3 Validity Syntax

abbreviation *validity-in* :: o⇒i⇒*bool* ([- *in* -] [*1*]) **where**
validity-in ≡ λ φ *v* . *v* ⊨ φ
definition *actual-validity* :: o⇒*bool* ([-] [*1*]) **where**
actual-validity ≡ λ φ . *dw* ⊨ φ
definition *necessary-validity* :: o⇒*bool* (□[-] [*1*]) **where**
necessary-validity ≡ λ φ . ∀ *v* . (*v* ⊨ φ)

3 General Quantification

Remark 13. *In order to define general quantifiers that can act on individuals as well as relations a type class is introduced which assumes the semantics of the all quantifier. This type class is then instantiated for individuals and relations.*

3.1 Type Class

class *quantifiable* = **fixes** *forall* :: ('*a*⇒o)⇒o (**binder** ∀ [*8*] *9*)
assumes *quantifiable-T8*: (*w* ⊨ (∀ *x* . ψ *x*)) = (∀ *x* . (*w* ⊨ (ψ *x*)))
begin
end

lemma (**in** *Semantics*) *T8*: **shows** (*w* ⊨ ∀ *x* . ψ *x*) = (∀ *x* . (*w* ⊨ ψ *x*))
using *quantifiable-T8* **.**

3.2 Instantiations

instantiation ν :: *quantifiable*
begin
definition *forall-*ν :: (ν⇒o)⇒o **where** *forall-*ν ≡ $forall_\nu$
instance proof
fix *w* :: *i* **and** ψ :: ν⇒o
show (*w* ⊨ ∀ *x*. ψ *x*) = (∀ *x*. (*w* ⊨ ψ *x*))
unfolding *forall-*ν*-def* **using** *Semantics.T8-*ν **.**
qed
end

instantiation o :: *quantifiable*
begin
definition *forall-*o :: (o⇒o)⇒o **where** *forall-*o ≡ $forall_o$
instance proof
fix *w* :: *i* **and** ψ :: o⇒o
show (*w* ⊨ ∀ *x*. ψ *x*) = (∀ *x*. (*w* ⊨ ψ *x*))
unfolding *forall-*o*-def* **using** *Semantics.T8-*o **.**
qed
end

instantiation Π_1 :: *quantifiable*
begin
definition *forall-*Π_1 :: (Π_1⇒o)⇒o **where** *forall-*Π_1 ≡ $forall_1$
instance proof
fix *w* :: *i* **and** ψ :: Π_1⇒o
show (*w* ⊨ ∀ *x*. ψ *x*) = (∀ *x*. (*w* ⊨ ψ *x*))
unfolding *forall-*Π_1*-def* **using** *Semantics.T8-1* **.**
qed
end

instantiation Π_2 :: *quantifiable*
begin
definition *forall-*Π_2 :: (Π_2⇒o)⇒o **where** *forall-*Π_2 ≡ $forall_2$
instance proof

fix w :: i **and** ψ :: $\Pi_2 \Rightarrow o$
show $(w \models \forall\, x.\ \psi\ x) = (\forall\, x.\ (w \models \psi\ x))$
unfolding *forall-*Π_2*-def* **using** *Semantics.T8-2* **.**
qed
end

instantiation Π_3 :: *quantifiable*
begin
definition *forall-*Π_3 :: $(\Pi_3 \Rightarrow o) \Rightarrow o$ **where** *forall-*$\Pi_3 \equiv$ *forall*$_3$
instance proof
fix w :: i **and** ψ :: $\Pi_3 \Rightarrow o$
show $(w \models \forall\, x.\ \psi\ x) = (\forall\, x.\ (w \models \psi\ x))$
unfolding *forall-*Π_3*-def* **using** *Semantics.T8-3* **.**
qed
end

4 Basic Definitions

4.1 Derived Connectives

definition *conj*::$o \Rightarrow o \Rightarrow o$ (**infixl &** *53*) **where**
conj $\equiv \lambda\ x\ y\ .\ \neg(x \rightarrow \neg y)$
definition *disj*::$o \Rightarrow o \Rightarrow o$ (**infixl** $\vee$ *52*) **where**
disj $\equiv \lambda\ x\ y\ .\ \neg x \rightarrow y$
definition *equiv*::$o \Rightarrow o \Rightarrow o$ (**infixl** $\equiv$ *51*) **where**
equiv $\equiv \lambda\ x\ y\ .\ (x \rightarrow y)$ **&** $(y \rightarrow x)$
definition *diamond*::$o \Rightarrow o$ ($\Diamond$- [*62*] *63*) **where**
diamond $\equiv \lambda\ \varphi\ .\ \neg\Box\neg\varphi$
definition (**in** *quantifiable*) *exists* :: $('a \Rightarrow o) \Rightarrow o$ (**binder** $\exists$ [*8*] *9*) **where**
exists $\equiv \lambda\ \varphi\ .\ \neg(\forall\ \ x\ .\ \neg\varphi\ x)$

named-theorems *conn-defs*
declare *diamond-def*[*conn-defs*] *conj-def*[*conn-defs*]
disj-def[*conn-defs*] *equiv-def*[*conn-defs*]
exists-def[*conn-defs*]

4.2 Abstract and Ordinary Objects

definition *Ordinary* :: Π_1 ($O!$) **where** *Ordinary* $\equiv \boldsymbol{\lambda} x.\ \Diamond(\!|E!,x^P|\!)$
definition *Abstract* :: Π_1 ($A!$) **where** *Abstract* $\equiv \boldsymbol{\lambda} x.\ \neg\Diamond(\!|E!,x^P|\!)$

4.3 Identity Definitions

definition *basic-identity*$_E$::Π_2 **where**
basic-identity$_E$ $\equiv \boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ (\!|O!,x^P|\!)$ **&** $(\!|O!,y^P|\!)$
& $\Box(\forall\ \ F.\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!)))$

definition *basic-identity*$_E$*-infix*::$\kappa \Rightarrow \kappa \Rightarrow o$ (**infixl** $=_E$ *63*) **where**
$x =_E\ y \equiv (\!|$*basic-identity*$_E,\ x,\ y|\!)$

definition *basic-identity*$_\kappa$ (**infixl** $=_\kappa$ *63*) **where**
basic-identity$_\kappa$ $\equiv \lambda\ x\ y\ .\ (x =_E\ y) \vee (\!|A!,x|\!)$ **&** $(\!|A!,y|\!)$
& $\Box(\forall\ \ F.\ \{\!|x,F|\!\} \equiv \{\!|y,F|\!\})$

definition *basic-identity*$_1$ (**infixl** $=_1$ *63*) **where**
basic-identity$_1$ $\equiv \lambda\ F\ G\ .\ \Box(\forall\ \ x.\ \{\!|x^P,F|\!\} \equiv \{\!|x^P,G|\!\})$

definition *basic-identity*$_2$ :: $\Pi_2 \Rightarrow \Pi_2 \Rightarrow o$ (**infixl** $=_2$ *63*) **where**
basic-identity$_2$ $\equiv \lambda\ F\ G\ .\ \ \forall\ \ x.\ ((\boldsymbol{\lambda} y.\ (\!|F,x^P,y^P|\!)) =_1 (\boldsymbol{\lambda} y.\ (\!|G,x^P,y^P|\!)))$
& $((\boldsymbol{\lambda} y.\ (\!|F,y^P,x^P|\!)) =_1 (\boldsymbol{\lambda} y.\ (\!|G,y^P,x^P|\!)))$

definition *basic-identity*$_3$::$\Pi_3 \Rightarrow \Pi_3 \Rightarrow$o (**infixl** $=_3$ *63*) **where**
basic-identity$_3$ $\equiv \lambda\ F\ G\ .\ \forall\ x\ y.\ (\boldsymbol{\lambda} z.\ (\!| F,z^P,x^P,y^P |\!)) =_1 (\boldsymbol{\lambda} z.\ (\!| G,z^P,x^P,y^P |\!))$
$\quad\ \&\ (\boldsymbol{\lambda} z.\ (\!| F,x^P,z^P,y^P |\!)) =_1 (\boldsymbol{\lambda} z.\ (\!| G,x^P,z^P,y^P |\!))$
$\quad\ \&\ (\boldsymbol{\lambda} z.\ (\!| F,x^P,y^P,z^P |\!)) =_1 (\boldsymbol{\lambda} z.\ (\!| G,x^P,y^P,z^P |\!))$

definition *basic-identity*$_0$::o$\Rightarrow$o$\Rightarrow$o (**infixl** $=_0$ *63*) **where**
basic-identity$_0$ $\equiv \lambda\ F\ G\ .\ (\boldsymbol{\lambda} y.\ F) =_1 (\boldsymbol{\lambda} y.\ G)$

5 MetaSolver

Remark 14. *meta-solver is a resolution prover that translates expressions in the embedded logic to expressions in the meta-logic, resp. semantic expressions. The rules for connectives, quantifiers, exemplification and encoding are straightforward. Furthermore, rules for the defined identities are derived. The defined identities in the embedded logic coincide with the meta-logical equality.*

locale *MetaSolver*
begin
interpretation *Semantics* **.**

named-theorems *meta-intro*
named-theorems *meta-elim*
named-theorems *meta-subst*
named-theorems *meta-cong*

method *meta-solver* = (*assumption* | *rule meta-intro*
| *erule meta-elim* | *drule meta-elim* | *subst meta-subst*
| *subst* (*asm*) *meta-subst* | (*erule notE*; (*meta-solver*; *fail*))
)+

5.1 Rules for Implication

lemma *ImplI*[*meta-intro*]: $([\varphi\ in\ v] \Longrightarrow [\psi\ in\ v]) \Longrightarrow ([\varphi \rightarrow \psi\ in\ v])$
by (*simp add*: *Semantics.T5*)
lemma *ImplE*[*meta-elim*]: $([\varphi \rightarrow \psi\ in\ v]) \Longrightarrow ([\varphi\ in\ v] \longrightarrow [\psi\ in\ v])$
by (*simp add*: *Semantics.T5*)
lemma *ImplS*[*meta-subst*]: $([\varphi \rightarrow \psi\ in\ v]) = ([\varphi\ in\ v] \longrightarrow [\psi\ in\ v])$
by (*simp add*: *Semantics.T5*)

5.2 Rules for Negation

lemma *NotI*[*meta-intro*]: $\neg[\varphi\ in\ v] \Longrightarrow [\neg\varphi\ in\ v]$
by (*simp add*: *Semantics.T4*)
lemma *NotE*[*meta-elim*]: $[\neg\varphi\ in\ v] \Longrightarrow \neg[\varphi\ in\ v]$
by (*simp add*: *Semantics.T4*)
lemma *NotS*[*meta-subst*]: $[\neg\varphi\ in\ v] = (\neg[\varphi\ in\ v])$
by (*simp add*: *Semantics.T4*)

5.3 Rules for Conjunction

lemma *ConjI*[*meta-intro*]: $([\varphi\ in\ v] \wedge [\psi\ in\ v]) \Longrightarrow [\varphi\ \&\ \psi\ in\ v]$
by (*simp add*: *conj-def NotS ImplS*)
lemma *ConjE*[*meta-elim*]: $[\varphi\ \&\ \psi\ in\ v] \Longrightarrow ([\varphi\ in\ v] \wedge [\psi\ in\ v])$
by (*simp add*: *conj-def NotS ImplS*)
lemma *ConjS*[*meta-subst*]: $[\varphi\ \&\ \psi\ in\ v] = ([\varphi\ in\ v] \wedge [\psi\ in\ v])$
by (*simp add*: *conj-def NotS ImplS*)

5.4 Rules for Equivalence

lemma *EquivI*[*meta-intro*]: $([\varphi\ in\ v] \longleftrightarrow [\psi\ in\ v]) \Longrightarrow [\varphi \equiv \psi\ in\ v]$
by (*simp add*: *equiv-def NotS ImplS ConjS*)
lemma *EquivE*[*meta-elim*]: $[\varphi \equiv \psi\ in\ v] \Longrightarrow ([\varphi\ in\ v] \longleftrightarrow [\psi\ in\ v])$
by (*auto simp*: *equiv-def NotS ImplS ConjS*)
lemma *EquivS*[*meta-subst*]: $[\varphi \equiv \psi\ in\ v] = ([\varphi\ in\ v] \longleftrightarrow [\psi\ in\ v])$
by (*auto simp*: *equiv-def NotS ImplS ConjS*)

5.5 Rules for Disjunction

lemma *DisjI*[*meta-intro*]: $([\varphi\ in\ v] \vee [\psi\ in\ v]) \Longrightarrow [\varphi \vee \psi\ in\ v]$
by (*auto simp*: *disj-def NotS ImplS*)
lemma *DisjE*[*meta-elim*]: $[\varphi \vee \psi\ in\ v] \Longrightarrow ([\varphi\ in\ v] \vee [\psi\ in\ v])$
by (*auto simp*: *disj-def NotS ImplS*)
lemma *DisjS*[*meta-subst*]: $[\varphi \vee \psi\ in\ v] = ([\varphi\ in\ v] \vee [\psi\ in\ v])$
by (*auto simp*: *disj-def NotS ImplS*)

5.6 Rules for Necessity

lemma *BoxI*[*meta-intro*]: $(\bigwedge v.[\varphi\ in\ v]) \Longrightarrow [\Box\varphi\ in\ v]$
by (*simp add*: *Semantics.T6*)
lemma *BoxE*[*meta-elim*]: $[\Box\varphi\ in\ v] \Longrightarrow (\bigwedge v.[\varphi\ in\ v])$
by (*simp add*: *Semantics.T6*)
lemma *BoxS*[*meta-subst*]: $[\Box\varphi\ in\ v] = (\forall\ v.[\varphi\ in\ v])$
by (*simp add*: *Semantics.T6*)

5.7 Rules for Possibility

lemma *DiaI*[*meta-intro*]: $(\exists\, v.[\varphi\ in\ v]) \Longrightarrow [\Diamond\varphi\ in\ v]$
by (*metis BoxS NotS diamond-def*)
lemma *DiaE*[*meta-elim*]: $[\Diamond\varphi\ in\ v] \Longrightarrow (\exists\, v.[\varphi\ in\ v])$
by (*metis BoxS NotS diamond-def*)
lemma *DiaS*[*meta-subst*]: $[\Diamond\varphi\ in\ v] = (\exists\ v.[\varphi\ in\ v])$
by (*metis BoxS NotS diamond-def*)

5.8 Rules for Quantification

lemma *AllI*[*meta-intro*]: $(\bigwedge x.\ [\varphi\ x\ in\ v]) \Longrightarrow [\forall\ x.\ \varphi\ x\ in\ v]$
by (*auto simp*: *T8*)
lemma *AllE*[*meta-elim*]: $[\forall\, x.\ \varphi\ x\ in\ v] \Longrightarrow (\bigwedge x.[\varphi\ x\ in\ v])$
by (*auto simp*: *T8*)
lemma *AllS*[*meta-subst*]: $[\forall\, x.\ \varphi\ x\ in\ v] = (\forall\, x.[\varphi\ x\ in\ v])$
by (*auto simp*: *T8*)

5.8.1 Rules for Existence

lemma *ExIRule*: $([\varphi\ y\ in\ v]) \Longrightarrow [\exists\, x.\ \varphi\ x\ in\ v]$
by (*auto simp*: *exists-def Semantics.T8 Semantics.T4*)
lemma *ExI*[*meta-intro*]: $(\exists\ y\ .\ [\varphi\ y\ in\ v]) \Longrightarrow [\exists\, x.\ \varphi\ x\ in\ v]$
by (*auto simp*: *exists-def Semantics.T8 Semantics.T4*)
lemma *ExE*[*meta-elim*]: $[\exists\, x.\ \varphi\ x\ in\ v] \Longrightarrow (\exists\ y\ .\ [\varphi\ y\ in\ v])$
by (*auto simp*: *exists-def Semantics.T8 Semantics.T4*)
lemma *ExS*[*meta-subst*]: $[\exists\, x.\ \varphi\ x\ in\ v] = (\exists\ y\ .\ [\varphi\ y\ in\ v])$
by (*auto simp*: *exists-def Semantics.T8 Semantics.T4*)
lemma *ExERule*: **assumes** $[\exists\, x.\ \varphi\ x\ in\ v]$ **obtains** x **where** $[\varphi\ x\ in\ v]$
using *ExE assms* **by** *auto*

5.9 Rules for Actuality

lemma *ActualI*[*meta-intro*]: $[\varphi\ in\ dw] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi\ in\ v]$

by (*auto simp*: *Semantics.T7*)
lemma *ActualE*[*meta-elim*]: $[\boldsymbol{\mathcal{A}}\varphi \ in \ v] \Longrightarrow [\varphi \ in \ dw]$
by (*auto simp*: *Semantics.T7*)
lemma *ActualS*[*meta-subst*]: $[\boldsymbol{\mathcal{A}}\varphi \ in \ v] = [\varphi \ in \ dw]$
by (*auto simp*: *Semantics.T7*)

5.10 Rules for Encoding

lemma *EncI*[*meta-intro*]:
assumes $\exists \ r \ o_1 \ . \ Some \ r = d_1 \ F \wedge Some \ o_1 = d_\kappa \ x \wedge o_1 \in en \ r$
shows $[\{\!|x,F|\!\} \ in \ v]$
using *assms* **by** (*auto simp*: *Semantics.T2*)
lemma *EncE*[*meta-elim*]:
assumes $[\{\!|x,F|\!\} \ in \ v]$
shows $\exists \ r \ o_1 \ . \ Some \ r = d_1 \ F \wedge Some \ o_1 = d_\kappa \ x \wedge o_1 \in en \ r$
using *assms* **by** (*auto simp*: *Semantics.T2*)
lemma *EncS*[*meta-subst*]:
$[\{\!|x,F|\!\} \ in \ v] = (\exists \ r \ o_1 \ . \ Some \ r = d_1 \ F \wedge Some \ o_1 = d_\kappa \ x \wedge o_1 \in en \ r)$
by (*auto simp*: *Semantics.T2*)

5.11 Rules for Exemplification

5.11.1 Zero-place Relations

lemma *Exe0I*[*meta-intro*]:
assumes $\exists \ r \ . \ Some \ r = d_0 \ p \wedge ex0 \ r \ v$
shows $[(\!|p|\!) \ in \ v]$
using *assms* **by** (*auto simp*: *Semantics.T3*)
lemma *Exe0E*[*meta-elim*]:
assumes $[(\!|p|\!) \ in \ v]$
shows $\exists \ r \ . \ Some \ r = d_0 \ p \wedge ex0 \ r \ v$
using *assms* **by** (*auto simp*: *Semantics.T3*)
lemma *Exe0S*[*meta-subst*]:
$[(\!|p|\!) \ in \ v] = (\exists \ r \ . \ Some \ r = d_0 \ p \wedge ex0 \ r \ v)$
by (*auto simp*: *Semantics.T3*)

5.11.2 One-Place Relations

lemma *Exe1I*[*meta-intro*]:
assumes $\exists \ r \ o_1 \ . \ Some \ r = d_1 \ F \wedge Some \ o_1 = d_\kappa \ x \wedge o_1 \in ex1 \ r \ v$
shows $[(\!|F,x|\!) \ in \ v]$
using *assms* **by** (*auto simp*: *Semantics.T1-1*)
lemma *Exe1E*[*meta-elim*]:
assumes $[(\!|F,x|\!) \ in \ v]$
shows $\exists \ r \ o_1 \ . \ Some \ r = d_1 \ F \wedge Some \ o_1 = d_\kappa \ x \wedge o_1 \in ex1 \ r \ v$
using *assms* **by** (*auto simp*: *Semantics.T1-1*)
lemma *Exe1S*[*meta-subst*]:
$[(\!|F,x|\!) \ in \ v] = (\exists \ r \ o_1 \ . \ Some \ r = d_1 \ F \wedge Some \ o_1 = d_\kappa \ x \wedge o_1 \in ex1 \ r \ v)$
by (*auto simp*: *Semantics.T1-1*)

5.11.3 Two-Place Relations

lemma *Exe2I*[*meta-intro*]:
assumes $\exists \ r \ o_1 \ o_2 \ . \ Some \ r = d_2 \ F \wedge Some \ o_1 = d_\kappa \ x$
$\wedge \ Some \ o_2 = d_\kappa \ y \wedge (o_1, \ o_2) \in ex2 \ r \ v$
shows $[(\!|F,x,y|\!) \ in \ v]$
using *assms* **by** (*auto simp*: *Semantics.T1-2*)
lemma *Exe2E*[*meta-elim*]:
assumes $[(\!|F,x,y|\!) \ in \ v]$
shows $\exists \ r \ o_1 \ o_2 \ . \ Some \ r = d_2 \ F \wedge Some \ o_1 = d_\kappa \ x$
$\wedge \ Some \ o_2 = d_\kappa \ y \wedge (o_1, \ o_2) \in ex2 \ r \ v$
using *assms* **by** (*auto simp*: *Semantics.T1-2*)
lemma *Exe2S*[*meta-subst*]:

$[\llparenthesis F,x,y \rrparenthesis \; in \; v] = (\exists \; r \; o_1 \; o_2 \; . \; Some \; r = d_2 \; F \wedge Some \; o_1 = d_\kappa \; x$
$\wedge \; Some \; o_2 = d_\kappa \; y \wedge (o_1, \; o_2) \in ex2 \; r \; v)$
by (*auto simp*: *Semantics.T1-2*)

5.11.4 Three-Place Relations

lemma *Exe3I*[*meta-intro*]:
assumes $\exists \; r \; o_1 \; o_2 \; o_3 \; . \; Some \; r = d_3 \; F \wedge Some \; o_1 = d_\kappa \; x$
$\wedge \; Some \; o_2 = d_\kappa \; y \wedge Some \; o_3 = d_\kappa \; z$
$\wedge \; (o_1, \; o_2, \; o_3) \in ex3 \; r \; v$
shows $[\llparenthesis F,x,y,z \rrparenthesis \; in \; v]$
using *assms* **by** (*auto simp*: *Semantics.T1-3*)
lemma *Exe3E*[*meta-elim*]:
assumes $[\llparenthesis F,x,y,z \rrparenthesis \; in \; v]$
shows $\exists \; r \; o_1 \; o_2 \; o_3 \; . \; Some \; r = d_3 \; F \wedge Some \; o_1 = d_\kappa \; x$
$\wedge \; Some \; o_2 = d_\kappa \; y \wedge Some \; o_3 = d_\kappa \; z$
$\wedge \; (o_1, \; o_2, \; o_3) \in ex3 \; r \; v$
using *assms* **by** (*auto simp*: *Semantics.T1-3*)
lemma *Exe3S*[*meta-subst*]:
$[\llparenthesis F,x,y,z \rrparenthesis \; in \; v] = (\exists \; r \; o_1 \; o_2 \; o_3 \; . \; Some \; r = d_3 \; F \wedge Some \; o_1 = d_\kappa \; x$
$\wedge \; Some \; o_2 = d_\kappa \; y \wedge Some \; o_3 = d_\kappa \; z$
$\wedge \; (o_1, \; o_2, \; o_3) \in ex3 \; r \; v)$
by (*auto simp*: *Semantics.T1-3*)

5.12 Rules for Being Ordinary

lemma *OrdI*[*meta-intro*]:
assumes $\exists \; o_1 \; y. \; Some \; o_1 = d_\kappa \; x \wedge o_1 = \omega\nu \; y$
shows $[\llparenthesis O!,x \rrparenthesis \; in \; v]$
proof −
have *IsProperInX* $(\lambda x. \; \Diamond \llparenthesis E!,x \rrparenthesis)$
by *show-proper*
moreover have $[\Diamond \llparenthesis E!,x \rrparenthesis \; in \; v]$
apply *meta-solver*
using *ConcretenessSemantics1* $propex_1$ *assms* **by** *fast*
ultimately show $[\llparenthesis O!,x \rrparenthesis \; in \; v]$
unfolding *Ordinary-def*
using *D5-1* $propex_1$ *assms ConcretenessSemantics1 Exe1S*
by *blast*
qed
lemma *OrdE*[*meta-elim*]:
assumes $[\llparenthesis O!,x \rrparenthesis \; in \; v]$
shows $\exists \; o_1 \; y. \; Some \; o_1 = d_\kappa \; x \wedge o_1 = \omega\nu \; y$
proof −
have $\exists r \; o_1. \; Some \; r = d_1 \; O! \wedge Some \; o_1 = d_\kappa \; x \wedge o_1 \in ex1 \; r \; v$
using *assms Exe1E* **by** *simp*
moreover have *IsProperInX* $(\lambda x. \; \Diamond \llparenthesis E!,x \rrparenthesis)$
by *show-proper*
ultimately have $[\Diamond \llparenthesis E!,x \rrparenthesis \; in \; v]$
using *D5-1* **unfolding** *Ordinary-def* **by** *fast*
thus *?thesis*
apply − **apply** *meta-solver*
using *ConcretenessSemantics2* **by** *blast*
qed
lemma *OrdS*[*meta-cong*]:
$[\llparenthesis O!,x \rrparenthesis \; in \; v] = (\exists \; o_1 \; y. \; Some \; o_1 = d_\kappa \; x \wedge o_1 = \omega\nu \; y)$
using *OrdI OrdE* **by** *blast*

5.13 Rules for Being Abstract

lemma *AbsI*[*meta-intro*]:
assumes $\exists \; o_1 \; y. \; Some \; o_1 = d_\kappa \; x \wedge o_1 = \alpha\nu \; y$
shows $[\llparenthesis A!,x \rrparenthesis \; in \; v]$

```
  proof -
    have IsProperInX (λx. ¬◊(|E!,x|))
      by show-proper
    moreover have [¬◊(|E!,x|) in v]
      apply meta-solver
      using ConcretenessSemantics2 propex1 assms
      by (metis ν.distinct(1) option.sel)
    ultimately show [(|A!,x|) in v]
      unfolding Abstract-def
      using D5-1 propex1 assms ConcretenessSemantics1 Exe1S
      by blast
  qed
lemma AbsE[meta-elim]:
  assumes [(|A!,x|) in v]
  shows ∃ o1 y. Some o1 = dκ x ∧ o1 = αν y
  proof -
    have 1: IsProperInX (λx. ¬◊(|E!,x|))
      by show-proper
    have ∃ r o1. Some r = d1 A! ∧ Some o1 = dκ x ∧ o1 ∈ ex1 r v
      using assms Exe1E by simp
    moreover hence [¬◊(|E!,x|) in v]
      using D5-1[OF 1]
      unfolding Abstract-def by fast
    ultimately show ?thesis
      apply - apply meta-solver
      using ConcretenessSemantics1 propex1
      by (metis ν.exhaust)
  qed
lemma AbsS[meta-cong]:
  [(|A!,x|) in v] = (∃ o1 y. Some o1 = dκ x ∧ o1 = αν y)
  using AbsI AbsE by blast
```

5.14 Rules for Definite Descriptions

```
lemma TheEqI:
  assumes ⋀x. [φ x in dw] = [ψ x in dw]
  shows (ιx. φ x) = (ιx. ψ x)
  proof -
    have 1: dκ (ιx. φ x) = dκ (ιx. ψ x)
      using assms D3 unfolding w0-def by simp
    {
      assume ∃ o1 . Some o1 = dκ (ιx. φ x)
      hence ?thesis using 1 dκ-inject by force
    }
    moreover {
      assume ¬(∃ o1 . Some o1 = dκ (ιx. φ x))
      hence ?thesis using 1 D3
      by (metis dκ.rep-eq evalκ-inverse)
    }
    ultimately show ?thesis by blast
  qed
```

5.15 Rules for Identity

5.15.1 Ordinary Objects

```
lemma EqEI[meta-intro]:
  assumes ∃ o1 o2. Some (ων o1) = dκ x ∧ Some (ων o2) = dκ y ∧ o1 = o2
  shows [x =E y in v]
  proof -
    obtain o1 o2 where 1:
      Some (ων o1) = dκ x ∧ Some (ων o2) = dκ y ∧ o1 = o2
```

using *assms* **by** *auto*
obtain r **where** *2*:
Some $r = d_2$ *basic-identity*$_E$
using *propex*$_2$ **by** *auto*
have $[(|O!,x|)$ **&** $(|O!,y|)$ **&** $\Box(\forall F.\ (|F,x|) \equiv (|F,y|))$ *in* $v]$
proof $-$
have $[(|O!,x|)$ *in* $v] \wedge [(|O!,y|)$ *in* $v]$
using *OrdI 1* **by** *blast*
moreover have $[\Box(\forall F.\ (|F,x|) \equiv (|F,y|))$ *in* $v]$
apply *meta-solver* **using** *1* **by** *force*
ultimately show *?thesis* **using** *ConjI* **by** *simp*
qed
moreover have *IsProperInXY* $(\lambda\ x\ y\ .\ (|O!,x|)$ **&** $(|O!,y|)$ **&** $\Box(\forall F.\ (|F,x|) \equiv (|F,y|)))$
by *show-proper*
ultimately have $(\omega\nu\ o_1,\ \omega\nu\ o_2) \in$ *ex2* $r\ v$
using *D5-2 1 2*
unfolding *basic-identity*$_E$*-def* **by** *fast*
thus $[x =_E y$ *in* $v]$
using *Exe2I 1 2*
unfolding *basic-identity*$_E$*-infix-def basic-identity*$_E$*-def*
by *blast*
qed
lemma *Eq*$_E$*E*[*meta-elim*]:
assumes $[x =_E y$ *in* $v]$
shows $\exists\ o_1\ o_2.$ *Some* $(\omega\nu\ o_1) = d_\kappa\ x \wedge$ *Some* $(\omega\nu\ o_2) = d_\kappa\ y \wedge o_1 = o_2$
proof $-$
have *IsProperInXY* $(\lambda\ x\ y\ .\ (|O!,x|)$ **&** $(|O!,y|)$ **&** $\Box(\forall F.\ (|F,x|) \equiv (|F,y|)))$
by *show-proper*
hence *1*: $[(|O!,x|)$ **&** $(|O!,y|)$ **&** $\Box(\forall\ F.\ (|F,x|) \equiv (|F,y|))$ *in* $v]$
using *assms* **unfolding** *basic-identity*$_E$*-def basic-identity*$_E$*-infix-def*
using *D4-2 T1-2 D5-2* **by** *meson*
hence *2*: $\exists\ o_1\ o_2\ .$ *Some* $(\omega\nu\ o_1) = d_\kappa\ x$
$\wedge$ *Some* $(\omega\nu\ o_2) = d_\kappa\ y$
apply (*subst* (*asm*) *ConjS*)
apply (*subst* (*asm*) *ConjS*)
using *OrdE* **by** *auto*
then obtain $o_1\ o_2$ **where** *3*:
Some $(\omega\nu\ o_1) = d_\kappa\ x \wedge$ *Some* $(\omega\nu\ o_2) = d_\kappa\ y$
by *auto*
have $\exists\ r\ .$ *Some* $r = d_1\ (\boldsymbol{\lambda}\ z\ .$ *make*o $(\lambda\ w\ s\ .\ d_\kappa\ (z^P) =$ *Some* $(\omega\nu\ o_1)))$
using *propex*$_1$ **by** *auto*
then obtain r **where** *4*:
Some $r = d_1\ (\boldsymbol{\lambda}\ z\ .$ *make*o $(\lambda\ w\ s\ .\ d_\kappa\ (z^P) =$ *Some* $(\omega\nu\ o_1)))$
by *auto*
hence *5*: $r = (\lambda u\ s\ w.\ \exists\ x\ .\ \nu\upsilon\ x = u \wedge$ *Some* $x =$ *Some* $(\omega\nu\ o_1))$
unfolding *lambdabinder1-def* d_1*-def* d_κ*-proper*
apply *transfer*
by *simp*
have $[\Box(\forall\ F.\ (|F,x|) \equiv (|F,y|))$ *in* $v]$
using *1* **using** *ConjE* **by** *blast*
hence *6*: $\forall\ v\ F\ .\ [(|F,x|)$ *in* $v] \longleftrightarrow [(|F,y|)$ *in* $v]$
using *BoxE EquivE AllE* **by** *fast*
hence $\forall\ v\ .\ ((\omega\nu\ o_1) \in$ *ex1* $r\ v) = ((\omega\nu\ o_2) \in$ *ex1* $r\ v)$
using *2 4* **unfolding** *valid-in-def*
by (*metis 3 6* d_1*.rep-eq* d_κ*-inject* d_κ*-proper ex1-def eval*o*-inverse exe1.rep-eq*
mem-Collect-eq option.sel rep-proper-id $\nu\kappa$*-proper valid-in.abs-eq*)
moreover have $(\omega\nu\ o_1) \in$ *ex1* $r\ v$
unfolding *5 ex1-def* **by** *simp*
ultimately have $(\omega\nu\ o_2) \in$ *ex1* $r\ v$
by *auto*
hence $o_1 = o_2$ **unfolding** *5 ex1-def* **by** (*auto simp*: *meta-aux*)
thus *?thesis*
using *3* **by** *auto*

```
qed
lemma EqES[meta-subst]:
  [x =_E y in v] = (∃ o_1 o_2. Some (ων o_1) = d_κ x ∧ Some (ων o_2) = d_κ y
                               ∧ o_1 = o_2)
  using EqEI EqEE by blast
```

5.15.2 Individuals

```
lemma EqκI[meta-intro]:
  assumes ∃ o_1 o_2. Some o_1 = d_κ x ∧ Some o_2 = d_κ y ∧ o_1 = o_2
  shows [x =_κ y in v]
proof −
  have x = y using assms d_κ-inject by meson
  moreover have [x =_κ x in v]
    unfolding basic-identity_κ-def
    apply meta-solver
    by (metis (no-types, lifting) assms AbsI Exe1E ν.exhaust)
  ultimately show ?thesis by auto
qed
lemma Eqκ-prop:
  assumes [x =_κ y in v]
  shows [φ x in v] = [φ y in v]
proof −
  have [x =_E y ∨ (|A!,x|) & (|A!,y|) & □(∀ F. {|x,F|} ≡ {|y,F|}) in v]
    using assms unfolding basic-identity_κ-def by simp
  moreover {
    assume [x =_E y in v]
    hence (∃ o_1 o_2. Some o_1 = d_κ x ∧ Some o_2 = d_κ y ∧ o_1 = o_2)
      using EqEE by fast
  }
  moreover {
    assume 1: [(|A!,x|) & (|A!,y|) & □(∀ F. {|x,F|} ≡ {|y,F|}) in v]
    hence 2: (∃ o_1 o_2 X Y. Some o_1 = d_κ x ∧ Some o_2 = d_κ y
                             ∧ o_1 = αν X ∧ o_2 = αν Y)
      using AbsE ConjE by meson
    moreover then obtain o_1 o_2 X Y where 3:
      Some o_1 = d_κ x ∧ Some o_2 = d_κ y ∧ o_1 = αν X ∧ o_2 = αν Y
      by auto
    moreover have 4: [□(∀ F. {|x,F|} ≡ {|y,F|}) in v]
      using 1 ConjE by blast
    hence 6: ∀ v F . [{|x,F|} in v] ⟷ [{|y,F|} in v]
      using BoxE AllE EquivE by fast
    hence 7: ∀ v r. (∃ o_1. Some o_1 = d_κ x ∧ o_1 ∈ en r)
                  = (∃ o_1. Some o_1 = d_κ y ∧ o_1 ∈ en r)
      apply − apply meta-solver
      using propex_1 d_1-inject apply simp
      apply transfer by simp
    hence 8: ∀ r. (o_1 ∈ en r) = (o_2 ∈ en r)
      using 3 d_κ-inject d_κ-proper apply simp
      by (metis option.inject)
    hence ∀ r. (o_1 ∈ r) = (o_2 ∈ r)
      unfolding en-def using 3
      by (metis Collect-cong Collect-mem-eq ν.simps(6)
               mem-Collect-eq makeΠ_1-cases)
    hence (o_1 ∈ { x . o_1 = x }) = (o_2 ∈ { x . o_1 = x })
      by metis
    hence o_1 = o_2 by simp
    hence (∃ o_1 o_2. Some o_1 = d_κ x ∧ Some o_2 = d_κ y ∧ o_1 = o_2)
      using 3 by auto
  }
  ultimately have x = y
    using DisjS using Semantics.d_κ-inject by auto
  thus (v ⊨ (φ x)) = (v ⊨ (φ y)) by simp
```

qed
lemma $Eq\kappa E[meta\text{-}elim]$:
assumes $[x =_\kappa y\ in\ v]$
shows $\exists\ o_1\ o_2.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = o_2$
proof −
have $\forall\ \varphi\ .\ (v \models \varphi\ x) = (v \models \varphi\ y)$
using *assms Eqκ-prop* **by** *blast*
moreover obtain φ **where** φ*-prop*:
$\varphi = (\lambda\ \alpha\ .\ makeo\ (\lambda\ w\ s\ .\ (\exists\ o_1\ o_2.\ Some\ o_1 = d_\kappa\ x$
$\wedge\ Some\ o_2 = d_\kappa\ \alpha \wedge o_1 = o_2)))$
by *auto*
ultimately have $(v \models \varphi\ x) = (v \models \varphi\ y)$ **by** *metis*
moreover have $(v \models \varphi\ x)$
using *assms* **unfolding** φ*-prop basic-identity*$_\kappa$*-def*
by (*metis* (*mono-tags*, *lifting*) *AbsS ConjE DisjS*
$Eq_E S$ *valid-in.abs-eq*)
ultimately have $(v \models \varphi\ y)$ **by** *auto*
thus *?thesis*
unfolding φ*-prop*
by (*simp add*: *valid-in-def meta-aux*)
qed
lemma $Eq\kappa S[meta\text{-}subst]$:
$[x =_\kappa y\ in\ v] = (\exists\ o_1\ o_2.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = o_2)$
using *EqκI EqκE* **by** *blast*

5.15.3 One-Place Relations

lemma $Eq_1 I[meta\text{-}intro]$: $F = G \Longrightarrow [F =_1 G\ in\ v]$
unfolding *basic-identity*$_1$*-def*
apply (*rule BoxI*, *rule AllI*, *rule EquivI*)
by *simp*
lemma $Eq_1 E[meta\text{-}elim]$: $[F =_1 G\ in\ v] \Longrightarrow F = G$
unfolding *basic-identity*$_1$*-def*
apply (*drule BoxE*, *drule-tac* $x{=}(\alpha\nu\ \{\ F\ \})$ **in** *AllE*, *drule EquivE*)
apply (*simp add*: *Semantics.T2*)
unfolding *en-def* d_κ*-def* d_1*-def*
using *νκ-proper rep-proper-id*
by (*simp add*: *rep-def proper-def meta-aux νκ.rep-eq*)
lemma $Eq_1 S[meta\text{-}subst]$: $[F =_1 G\ in\ v] = (F = G)$
using $Eq_1 I$ $Eq_1 E$ **by** *auto*
lemma Eq_1*-prop*: $[F =_1 G\ in\ v] \Longrightarrow [\varphi\ F\ in\ v] = [\varphi\ G\ in\ v]$
using $Eq_1 E$ **by** *blast*

5.15.4 Two-Place Relations

lemma $Eq_2 I[meta\text{-}intro]$: $F = G \Longrightarrow [F =_2 G\ in\ v]$
unfolding *basic-identity*$_2$*-def*
apply (*rule AllI*, *rule ConjI*, (*subst* $Eq_1 S$)+)
by *simp*
lemma $Eq_2 E[meta\text{-}elim]$: $[F =_2 G\ in\ v] \Longrightarrow F = G$
proof −
assume $[F =_2 G\ in\ v]$
hence *1*: $[\forall\ x.\ (\boldsymbol{\lambda} y.\ (\!|F,x^P,y^P|\!)) =_1 (\boldsymbol{\lambda} y.\ (\!|G,x^P,y^P|\!))\ in\ v]$
unfolding *basic-identity*$_2$*-def*
apply − **apply** *meta-solver* **by** *auto*
{
fix *u v s w*
obtain x **where** x*-def*: $\nu\upsilon\ x = v$ **by** (*metis νυ-surj surj-def*)
obtain a **where** a*-def*:
$a = (\lambda u\ s\ w.\ \exists xa.\ \nu\upsilon\ xa = u \wedge eval\Pi_2\ F\ (\nu\upsilon\ x)\ (\nu\upsilon\ xa)\ s\ w)$
by *auto*
obtain b **where** b*-def*:
$b = (\lambda u\ s\ w.\ \exists xa.\ \nu\upsilon\ xa = u \wedge eval\Pi_2\ G\ (\nu\upsilon\ x)\ (\nu\upsilon\ xa)\ s\ w)$

by *auto*
have $a = b$ **unfolding** *a-def b-def*
using *1* **apply** $-$ **apply** *meta-solver*
by (*auto simp*: *meta-defs meta-aux make*Π_1*-inject*)
hence $a\ u\ s\ w = b\ u\ s\ w$ **by** *auto*
hence $(eval\Pi_2\ F\ (\nu\upsilon\ x)\ u\ s\ w) = (eval\Pi_2\ G\ (\nu\upsilon\ x)\ u\ s\ w)$
unfolding *a-def b-def*
by (*metis* (*no-types*, *hide-lams*) $\nu\upsilon$*-surj surj-def*)
hence $(eval\Pi_2\ F\ v\ u\ s\ w) = (eval\Pi_2\ G\ v\ u\ s\ w)$
unfolding *x-def* **by** *auto*
}
hence $(eval\Pi_2\ F) = (eval\Pi_2\ G)$ **by** *blast*
thus $F = G$ **by** (*simp add*: *eval*Π_2*-inject*)
qed
lemma *Eq*$_2$*S*[*meta-subst*]: $[F =_2 G\ in\ v] = (F = G)$
using *Eq*$_2$*I Eq*$_2$*E* **by** *auto*
lemma *Eq*$_2$*-prop*: $[F =_2 G\ in\ v] \Longrightarrow [\varphi\ F\ in\ v] = [\varphi\ G\ in\ v]$
using *Eq*$_2$*E* **by** *blast*

5.15.5 Three-Place Relations

lemma *Eq*$_3$*I*[*meta-intro*]: $F = G \Longrightarrow [F =_3 G\ in\ v]$
apply (*simp add*: *meta-defs meta-aux conn-defs forall-*ν*-def basic-identity*$_3$*-def*)
using *MetaSolver.Eq*$_1$*I valid-in.rep-eq* **by** *auto*
lemma *Eq*$_3$*E*[*meta-elim*]: $[F =_3 G\ in\ v] \Longrightarrow F = G$
proof $-$

assume $[F =_3 G\ in\ v]$
hence *1*: $[\forall\ x\ y.\ (\boldsymbol{\lambda} z.\ (\!|F,x^P,y^P,z^P|\!)) =_1 (\boldsymbol{\lambda} z.\ (\!|G,x^P,y^P,z^P|\!))\ in\ v]$
unfolding *basic-identity*$_3$*-def*
apply $-$ **apply** *meta-solver* **by** *auto*
{
fix $u\ v\ r\ s\ w$
obtain x **where** *x-def*: $\nu\upsilon\ x = v$ **by** (*metis* $\nu\upsilon$*-surj surj-def*)
obtain y **where** *y-def*: $\nu\upsilon\ y = r$ **by** (*metis* $\nu\upsilon$*-surj surj-def*)
obtain a **where** *a-def*:
$a = (\lambda u\ s\ w.\ \exists xa.\ \nu\upsilon\ xa = u \wedge eval\Pi_3\ F\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ (\nu\upsilon\ xa)\ s\ w)$
by *auto*
obtain b **where** *b-def*:
$b = (\lambda u\ s\ w.\ \exists xa.\ \nu\upsilon\ xa = u \wedge eval\Pi_3\ G\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ (\nu\upsilon\ xa)\ s\ w)$
by *auto*
have $a = b$ **unfolding** *a-def b-def*
using *1* **apply** $-$ **apply** *meta-solver*
by (*auto simp*: *meta-defs meta-aux make*Π_1*-inject*)
hence $a\ u\ s\ w = b\ u\ s\ w$ **by** *auto*
hence $(eval\Pi_3\ F\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ u\ s\ w) = (eval\Pi_3\ G\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ u\ s\ w)$
unfolding *a-def b-def*
by (*metis* (*no-types*, *hide-lams*) $\nu\upsilon$*-surj surj-def*)
hence $(eval\Pi_3\ F\ v\ r\ u\ s\ w) = (eval\Pi_3\ G\ v\ r\ u\ s\ w)$
unfolding *x-def y-def* **by** *auto*
}
hence $(eval\Pi_3\ F) = (eval\Pi_3\ G)$ **by** *blast*
thus $F = G$ **by** (*simp add*: *eval*Π_3*-inject*)
qed
lemma *Eq*$_3$*S*[*meta-subst*]: $[F =_3 G\ in\ v] = (F = G)$
using *Eq*$_3$*I Eq*$_3$*E* **by** *auto*
lemma *Eq*$_3$*-prop*: $[F =_3 G\ in\ v] \Longrightarrow [\varphi\ F\ in\ v] = [\varphi\ G\ in\ v]$
using *Eq*$_3$*E* **by** *blast*

5.15.6 Propositions

lemma *Eq*$_0$*I*[*meta-intro*]: $x = y \Longrightarrow [x =_0 y\ in\ v]$
unfolding *basic-identity*$_0$*-def* **by** (*simp add*: *Eq*$_1$*S*)

lemma Eq_0E[*meta-elim*]: $[F =_0 G \; in \; v] \Longrightarrow F = G$
proof −
assume $[F =_0 G \; in \; v]$
hence $[(\boldsymbol{\lambda} y. \; F) =_1 (\boldsymbol{\lambda} y. \; G) \; in \; v]$
unfolding *basic-identity*$_0$-*def* **by** *simp*
hence $(\boldsymbol{\lambda} y. \; F) = (\boldsymbol{\lambda} y. \; G)$
using Eq_1S **by** *simp*
hence $(\lambda u \; s \; w. \; (\exists \, x. \; \nu \upsilon \; x = u) \wedge evalo \; F \; s \; w)$
$= (\lambda u \; s \; w. \; (\exists \, x. \; \nu \upsilon \; x = u) \wedge evalo \; G \; s \; w)$
apply (*simp add*: *meta-defs meta-aux*)
by (*metis* (*no-types*, *lifting*) *UNIV-I make*Π_1-*inverse*)
hence $\bigwedge s \; w.(evalo \; F \; s \; w) = (evalo \; G \; s \; w)$
by *metis*
hence $(evalo \; F) = (evalo \; G)$ **by** *blast*
thus $F = G$
by (*metis evalo-inverse*)
qed
lemma Eq_0S[*meta-subst*]: $[F =_0 G \; in \; v] = (F = G)$
using Eq_0I Eq_0E **by** *auto*
lemma Eq_0-*prop*: $[F =_0 G \; in \; v] \Longrightarrow [\varphi \; F \; in \; v] = [\varphi \; G \; in \; v]$
using Eq_0E **by** *blast*

end

6 General Identity

Remark 15. *In order to define a general identity symbol that can act on all types of terms a type class is introduced which assumes the substitution property which is needed to derive the corresponding axiom. This type class is instantiated for all relation types, individual terms and individuals.*

6.1 Type Classes

class *identifiable* =
fixes *identity* :: $'a \Rightarrow 'a \Rightarrow$o (**infixl** = *63*)
assumes *l-identity*:
$w \models x = y \Longrightarrow w \models \varphi \; x \Longrightarrow w \models \varphi \; y$
begin
abbreviation *notequal* (**infixl** $\neq$ *63*) **where**
$notequal \equiv \lambda \; x \; y \; . \; \neg(x = y)$
end

class *quantifiable-and-identifiable* = *quantifiable* + *identifiable*
begin
definition *exists-unique*::($'a \Rightarrow$o)$\Rightarrow$o (**binder** $\exists!$ [*8*] *9*) **where**
$exists\text{-}unique \equiv \lambda \; \varphi \; . \; \exists \; \alpha \; . \; \varphi \; \alpha$ **&** $(\forall \, \beta. \; \varphi \; \beta \rightarrow \beta = \alpha)$

declare *exists-unique-def*[*conn-defs*]
end

6.2 Instantiations

instantiation κ :: *identifiable*
begin
definition *identity*-κ **where** *identity*-κ $\equiv$ *basic-identity*$_\kappa$
instance proof
fix $x \; y :: \kappa$ **and** $w \; \varphi$
show $[x = y \; in \; w] \Longrightarrow [\varphi \; x \; in \; w] \Longrightarrow [\varphi \; y \; in \; w]$
unfolding *identity*-κ-*def*

    using *MetaSolver.Eqκ-prop* **..**
  qed
end

instantiation ν :: *identifiable*
begin
  definition *identity-ν* **where** *identity-ν* $\equiv \lambda\ x\ y\ .\ x^P = y^P$
  instance proof
    fix $\alpha :: \nu$ **and** $\beta :: \nu$ **and** $v\ \varphi$
    assume $v \models \alpha = \beta$
    hence $v \models \alpha^P = \beta^P$
      unfolding *identity-ν-def* **by** *auto*
    hence $\bigwedge\varphi.(v \models \varphi\ (\alpha^P)) \Longrightarrow (v \models \varphi\ (\beta^P))$
      using *l-identity* **by** *auto*
    hence $(v \models \varphi\ (rep\ (\alpha^P))) \Longrightarrow (v \models \varphi\ (rep\ (\beta^P)))$
      by *meson*
    thus $(v \models \varphi\ \alpha) \Longrightarrow (v \models \varphi\ \beta)$
      by (*simp only*: *rep-proper-id*)
  qed
end

instantiation Π_1 :: *identifiable*
begin
  definition *identity*-Π_1 **where** *identity*-$\Pi_1 \equiv$ *basic-identity*$_1$
  instance proof
    fix $F\ G :: \Pi_1$ **and** $w\ \varphi$
    show $(w \models F = G) \Longrightarrow (w \models \varphi\ F) \Longrightarrow (w \models \varphi\ G)$
      unfolding *identity*-Π_1-*def* **using** *MetaSolver.Eq*$_1$*-prop* **..**
  qed
end

instantiation Π_2 :: *identifiable*
begin
  definition *identity*-Π_2 **where** *identity*-$\Pi_2 \equiv$ *basic-identity*$_2$
  instance proof
    fix $F\ G :: \Pi_2$ **and** $w\ \varphi$
    show $(w \models F = G) \Longrightarrow (w \models \varphi\ F) \Longrightarrow (w \models \varphi\ G)$
      unfolding *identity*-Π_2-*def* **using** *MetaSolver.Eq*$_2$*-prop* **..**
  qed
end

instantiation Π_3 :: *identifiable*
begin
  definition *identity*-Π_3 **where** *identity*-$\Pi_3 \equiv$ *basic-identity*$_3$
  instance proof
    fix $F\ G :: \Pi_3$ **and** $w\ \varphi$
    show $(w \models F = G) \Longrightarrow (w \models \varphi\ F) \Longrightarrow (w \models \varphi\ G)$
      unfolding *identity*-Π_3-*def* **using** *MetaSolver.Eq*$_3$*-prop* **..**
  qed
end

instantiation o :: *identifiable*
begin
  definition *identity*-o **where** *identity*-o $\equiv$ *basic-identity*$_0$
  instance proof
    fix $F\ G$ :: o **and** $w\ \varphi$
    show $(w \models F = G) \Longrightarrow (w \models \varphi\ F) \Longrightarrow (w \models \varphi\ G)$
      unfolding *identity*-o-*def* **using** *MetaSolver.Eq*$_0$*-prop* **..**
  qed
end

instance ν :: *quantifiable-and-identifiable* **..**
instance Π_1 :: *quantifiable-and-identifiable* **..**

instance Π_2 :: *quantifiable-and-identifiable* **..**
instance Π_3 :: *quantifiable-and-identifiable* **..**
instance o :: *quantifiable-and-identifiable* **..**

6.3 New Identity Definitions

Remark 16. *The basic definitions of identity use type specific quantifiers and identity symbols. Equivalent definitions that use the general identity symbol and general quantifiers are provided.*

named-theorems *identity-defs*
lemma $identity_E$*-def*[*identity-defs*]:
  $basic\text{-}identity_E \equiv \boldsymbol{\lambda}^2\ (\lambda x\ y.\ \langle\!|O!,x^P|\!\rangle\ \&\ \langle\!|O!,y^P|\!\rangle\ \&\ \Box(\forall F.\ \langle\!|F,x^P|\!\rangle \equiv \langle\!|F,y^P|\!\rangle))$
  unfolding $basic\text{-}identity_E$*-def forall*-Π_1*-def* **by** *simp*
lemma $identity_E$*-infix-def*[*identity-defs*]:
  $x =_E y \equiv \langle\!|basic\text{-}identity_E,x,y|\!\rangle$ **using** $basic\text{-}identity_E$*-infix-def* **.**
lemma $identity_\kappa$*-def*[*identity-defs*]:
  $op = \equiv \lambda x\ y.\ x =_E y \lor \langle\!|A!,x|\!\rangle\ \&\ \langle\!|A!,y|\!\rangle\ \&\ \Box(\forall\ F.\ \{\!|x,F|\!\} \equiv \{\!|y,F|\!\})$
  unfolding *identity*-κ*-def* $basic\text{-}identity_\kappa$*-def forall*-$\Pi_1$*-def* **by** *simp*
lemma $identity_\nu$*-def*[*identity-defs*]:
  $op = \equiv \lambda x\ y.\ (x^P) =_E (y^P) \lor \langle\!|A!,x^P|\!\rangle\ \&\ \langle\!|A!,y^P|\!\rangle\ \&\ \Box(\forall\ F.\ \{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\})$
  unfolding *identity*-ν*-def* $identity_\kappa$*-def* **by** *simp*
lemma $identity_1$*-def*[*identity-defs*]:
  $op = \equiv \lambda F\ G.\ \Box(\forall\ x\ .\ \{\!|x^P,F|\!\} \equiv \{\!|x^P,G|\!\})$
  unfolding *identity*-Π_1*-def* $basic\text{-}identity_1$*-def forall*-ν*-def* **by** *simp*
lemma $identity_2$*-def*[*identity-defs*]:
  $op = \equiv \lambda F\ G.\ \forall\ x.\ (\boldsymbol{\lambda} y.\ \langle\!|F,x^P,y^P|\!\rangle) = (\boldsymbol{\lambda} y.\ \langle\!|G,x^P,y^P|\!\rangle)$
    $\&\ (\boldsymbol{\lambda} y.\ \langle\!|F,y^P,x^P|\!\rangle) = (\boldsymbol{\lambda} y.\ \langle\!|G,y^P,x^P|\!\rangle)$
  unfolding *identity*-Π_2*-def identity*-Π_1*-def* $basic\text{-}identity_2$*-def forall*-ν*-def* **by** *simp*
lemma $identity_3$*-def*[*identity-defs*]:
  $op = \equiv \lambda F\ G.\ \forall\ x\ y.\ (\boldsymbol{\lambda} z.\ \langle\!|F,z^P,x^P,y^P|\!\rangle) = (\boldsymbol{\lambda} z.\ \langle\!|G,z^P,x^P,y^P|\!\rangle)$
    $\&\ (\boldsymbol{\lambda} z.\ \langle\!|F,x^P,z^P,y^P|\!\rangle) = (\boldsymbol{\lambda} z.\ \langle\!|G,x^P,z^P,y^P|\!\rangle)$
    $\&\ (\boldsymbol{\lambda} z.\ \langle\!|F,x^P,y^P,z^P|\!\rangle) = (\boldsymbol{\lambda} z.\ \langle\!|G,x^P,y^P,z^P|\!\rangle)$
  unfolding *identity*-Π_3*-def identity*-Π_1*-def* $basic\text{-}identity_3$*-def forall*-ν*-def* **by** *simp*
lemma $identity_o$*-def*[*identity-defs*]: $op = \equiv \lambda F\ G.\ (\boldsymbol{\lambda} y.\ F) = (\boldsymbol{\lambda} y.\ G)$
  unfolding *identity*-o*-def identity*-Π_1*-def* $basic\text{-}identity_0$*-def* **by** *simp*

7 The Axioms of PLM

Remark 17. *The axioms of PLM can now be derived from the Semantics and the model structure.*

locale *Axioms*
begin
  interpretation *MetaSolver* **.**
  interpretation *Semantics* **.**
  named-theorems *axiom*

Remark 18. *The special syntax* [[-]] *is introduced for stating the axioms. Modally-fragile axioms are stated with the syntax for actual validity* [-].

  definition *axiom* :: o⇒*bool* ([[-]]) **where** $axiom \equiv \lambda\ \varphi\ .\ \forall\ v\ .\ [\varphi\ in\ v]$

  method *axiom-meta-solver* = ((((*unfold axiom-def*)?, *rule allI*) | (*unfold actual-validity-def*)?), *meta-solver*,
    (*simp* | (*auto*; *fail*))?)

7.1 Closures

Remark 19. *Rules resembling the concepts of closures in PLM are derived. Theorem attributes are introduced to aid in the instantiation of the axioms.*

lemma *axiom-instance*[*axiom*]: $[[\varphi]] \Longrightarrow [\varphi\ in\ v]$
  unfolding *axiom-def* **by** *simp*
lemma *closures-universal*[*axiom*]: $(\bigwedge x.[[\varphi\ x]]) \Longrightarrow [[\forall\ x.\ \varphi\ x]]$
  by *axiom-meta-solver*
lemma *closures-actualization*[*axiom*]: $[[\varphi]] \Longrightarrow [[\boldsymbol{\mathcal{A}}\ \varphi]]$
  by *axiom-meta-solver*
lemma *closures-necessitation*[*axiom*]: $[[\varphi]] \Longrightarrow [[\Box\ \varphi]]$
  by *axiom-meta-solver*
lemma *necessitation-averse-axiom-instance*[*axiom*]: $[\varphi] \Longrightarrow [\varphi\ in\ dw]$
  by *axiom-meta-solver*
lemma *necessitation-averse-closures-universal*[*axiom*]: $(\bigwedge x.[\varphi\ x]) \Longrightarrow [\forall\ x.\ \varphi\ x]$
  by *axiom-meta-solver*

attribute-setup *axiom-instance* = ⟪
  Scan.succeed (*Thm.rule-attribute* []
    (*fn - => fn thm => thm RS* @{*thm axiom-instance*}))
⟫

attribute-setup *necessitation-averse-axiom-instance* = ⟪
  Scan.succeed (*Thm.rule-attribute* []
    (*fn - => fn thm => thm RS* @{*thm necessitation-averse-axiom-instance*}))
⟫

attribute-setup *axiom-necessitation* = ⟪
  Scan.succeed (*Thm.rule-attribute* []
    (*fn - => fn thm => thm RS* @{*thm closures-necessitation*}))
⟫

attribute-setup *axiom-actualization* = ⟪
  Scan.succeed (*Thm.rule-attribute* []
    (*fn - => fn thm => thm RS* @{*thm closures-actualization*}))
⟫

attribute-setup *axiom-universal* = ⟪
  Scan.succeed (*Thm.rule-attribute* []
    (*fn - => fn thm => thm RS* @{*thm closures-universal*}))
⟫

7.2 Axioms for Negations and Conditionals

lemma *pl-1*[*axiom*]:
  $[[\varphi \rightarrow (\psi \rightarrow \varphi)]]$
  by *axiom-meta-solver*
lemma *pl-2*[*axiom*]:
  $[[(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi))]]$
  by *axiom-meta-solver*
lemma *pl-3*[*axiom*]:
  $[[(\neg\varphi \rightarrow \neg\psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \varphi)]]$
  by *axiom-meta-solver*

7.3 Axioms of Identity

lemma *l-identity*[*axiom*]:
  $[[\alpha = \beta \rightarrow (\varphi\ \alpha \rightarrow \varphi\ \beta)]]$
  using *l-identity* **apply** − **by** *axiom-meta-solver*

7.4 Axioms of Quantification

lemma *cqt-1*[*axiom*]:
$[[(\forall\ \alpha.\ \varphi\ \alpha) \to \varphi\ \alpha]]$
by *axiom-meta-solver*
lemma *cqt-1-κ*[*axiom*]:
$[[(\forall\ \alpha.\ \varphi\ (\alpha^P)) \to ((\exists\ \beta\ .\ (\beta^P) = \alpha) \to \varphi\ \alpha)]]$
proof −
{
fix v
assume *1*: $[(\forall\ \alpha.\ \varphi\ (\alpha^P))\ in\ v]$
assume $[(\exists\ \beta\ .\ (\beta^P) = \alpha)\ in\ v]$
then obtain β **where** *2*:
$[(\beta^P) = \alpha\ in\ v]$ **by** (*rule ExERule*)
hence $[\varphi\ (\beta^P)\ in\ v]$ **using** *1 AllE* **by** *fast*
hence $[\varphi\ \alpha\ in\ v]$
using *l-identity*[**where** $\varphi{=}\varphi$, *axiom-instance*]
ImplS 2 **by** *simp*
}
thus $[[(\forall\ \alpha.\ \varphi\ (\alpha^P)) \to ((\exists\ \beta\ .\ (\beta^P) = \alpha) \to \varphi\ \alpha)]]$
unfolding *axiom-def* **using** *ImplI* **by** *blast*
qed
lemma *cqt-3*[*axiom*]:
$[[(\forall\alpha.\ \varphi\ \alpha \to \psi\ \alpha) \to ((\forall\alpha.\ \varphi\ \alpha) \to (\forall\alpha.\ \psi\ \alpha))]]$
by *axiom-meta-solver*
lemma *cqt-4*[*axiom*]:
$[[\varphi \to (\forall\alpha.\ \varphi)]]$
by *axiom-meta-solver*

inductive *SimpleExOrEnc*
where *SimpleExOrEnc* $(\lambda\ x\ .\ (\!|F,x|\!))$
| *SimpleExOrEnc* $(\lambda\ x\ .\ (\!|F,x,y|\!))$
| *SimpleExOrEnc* $(\lambda\ x\ .\ (\!|F,y,x|\!))$
| *SimpleExOrEnc* $(\lambda\ x\ .\ (\!|F,x,y,z|\!))$
| *SimpleExOrEnc* $(\lambda\ x\ .\ (\!|F,y,x,z|\!))$
| *SimpleExOrEnc* $(\lambda\ x\ .\ (\!|F,y,z,x|\!))$
| *SimpleExOrEnc* $(\lambda\ x\ .\ \{\!|x,F|\!\})$

lemma *cqt-5*[*axiom*]:
assumes *SimpleExOrEnc* ψ
shows $[[(\psi\ (\iota x\ .\ \varphi\ x)) \to (\exists\ \alpha.\ (\alpha^P) = (\iota x\ .\ \varphi\ x))]]$
proof −
have $\forall\ w\ .\ ([(\psi\ (\iota x\ .\ \varphi\ x))\ in\ w] \longrightarrow (\exists\ o_1\ .\ Some\ o_1 = d_\kappa\ (\iota x\ .\ \varphi\ x)))$
using *assms* **apply** *induct* **by** (*meta-solver*;*metis*)+
thus *?thesis*
apply − **unfolding** *identity-κ-def*
apply *axiom-meta-solver*
using d_κ*-proper* **by** *auto*
qed

lemma *cqt-5-mod*[*axiom*]:
assumes *SimpleExOrEnc* ψ
shows $[[\psi\ \tau \to (\exists\ \alpha\ .\ (\alpha^P) = \tau)]]$
proof −
have $\forall\ w\ .\ ([(\psi\ \tau)\ in\ w] \longrightarrow (\exists\ o_1\ .\ Some\ o_1 = d_\kappa\ \tau))$
using *assms* **apply** *induct* **by** (*meta-solver*;*metis*)+
thus *?thesis*
apply − **unfolding** *identity-κ-def*
apply *axiom-meta-solver*
using d_κ*-proper* **by** *auto*
qed

7.5 Axioms of Actuality

lemma *logic-actual*[*axiom*]: $[(\boldsymbol{\mathcal{A}}\varphi) \equiv \varphi]$
by *axiom-meta-solver*
lemma $[[(\boldsymbol{\mathcal{A}}\varphi) \equiv \varphi]]$
nitpick[*user-axioms*, *expect* = *genuine*, *card* = *1*, *card i* = *2*]
oops — Counter-model by nitpick

lemma *logic-actual-nec-1*[*axiom*]:
$[[\boldsymbol{\mathcal{A}}\neg\varphi \equiv \neg\boldsymbol{\mathcal{A}}\varphi]]$
by *axiom-meta-solver*
lemma *logic-actual-nec-2*[*axiom*]:
$[[(\boldsymbol{\mathcal{A}}(\varphi \rightarrow \psi)) \equiv (\boldsymbol{\mathcal{A}}\varphi \rightarrow \boldsymbol{\mathcal{A}}\psi)]]$
by *axiom-meta-solver*
lemma *logic-actual-nec-3*[*axiom*]:
$[[\boldsymbol{\mathcal{A}}(\forall\, \alpha.\ \varphi\ \alpha) \equiv (\forall\, \alpha.\ \boldsymbol{\mathcal{A}}(\varphi\ \alpha))]]$
by *axiom-meta-solver*
lemma *logic-actual-nec-4*[*axiom*]:
$[[\boldsymbol{\mathcal{A}}\varphi \equiv \boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}\varphi]]$
by *axiom-meta-solver*

7.6 Axioms of Necessity

lemma *qml-1*[*axiom*]:
$[[\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)]]$
by *axiom-meta-solver*
lemma *qml-2*[*axiom*]:
$[[\Box\varphi \rightarrow \varphi]]$
by *axiom-meta-solver*
lemma *qml-3*[*axiom*]:
$[[\Diamond\varphi \rightarrow \Box\Diamond\varphi]]$
by *axiom-meta-solver*
lemma *qml-4*[*axiom*]:
$[[\Diamond(\exists\, x.\ (|E!,x^P|) \ \&\ \Diamond\neg(|E!,x^P|)) \ \&\ \Diamond\neg(\exists\, x.\ (|E!,x^P|) \ \&\ \Diamond\neg(|E!,x^P|))]]$
unfolding *axiom-def*
using *PossiblyContingentObjectExistsAxiom*
PossiblyNoContingentObjectExistsAxiom
apply (*simp add*: *meta-defs meta-aux conn-defs forall-ν-def*
split: *ν.split υ.split*)
by (*metis $\nu\upsilon$-$\omega\nu$-is-$\omega\upsilon$ υ.distinct(1) υ.inject(1)*)

7.7 Axioms of Necessity and Actuality

lemma *qml-act-1*[*axiom*]:
$[[\boldsymbol{\mathcal{A}}\varphi \rightarrow \Box\boldsymbol{\mathcal{A}}\varphi]]$
by *axiom-meta-solver*
lemma *qml-act-2*[*axiom*]:
$[[\Box\varphi \equiv \boldsymbol{\mathcal{A}}(\Box\varphi)]]$
by *axiom-meta-solver*

7.8 Axioms of Descriptions

lemma *descriptions*[*axiom*]:
$[[x^P = (\boldsymbol{\iota}x.\ \varphi\ x) \equiv (\forall\, z.(\boldsymbol{\mathcal{A}}(\varphi\ z) \equiv z = x))]]$
unfolding *axiom-def*
proof (*rule allI*, *rule EquivI*; *rule*)
fix v
assume $[x^P = (\boldsymbol{\iota}x.\ \varphi\ x)\ in\ v]$
moreover hence *1*:
$\exists\, o_1\ o_2.\ Some\ o_1 = d_\kappa\ (x^P) \wedge Some\ o_2 = d_\kappa\ (\boldsymbol{\iota}x.\ \varphi\ x) \wedge o_1 = o_2$
apply − **unfolding** *identity-κ-def* **by** *meta-solver*
then obtain $o_1\ o_2$ **where** *2*:

$Some\ o_1 = d_\kappa\ (x^P) \wedge Some\ o_2 = d_\kappa\ (\iota x.\ \varphi\ x) \wedge o_1 = o_2$
by *auto*
hence *3*:
$(\exists\ x\ .((w_0 \models \varphi\ x) \wedge (\forall y.\ (w_0 \models \varphi\ y) \longrightarrow y = x)))$
$\wedge\ d_\kappa\ (\iota x.\ \varphi\ x) = Some\ (THE\ x.\ (w_0 \models \varphi\ x))$
using *D3* **by** (*metis option.distinct*(*1*))
then obtain X **where** *4*:
$((w_0 \models \varphi\ X) \wedge (\forall y.\ (w_0 \models \varphi\ y) \longrightarrow y = X))$
by *auto*
moreover have $o_1 = (THE\ x.\ (w_0 \models \varphi\ x))$
using *2 3* **by** *auto*
ultimately have *5*: $X = o_1$
by (*metis* (*mono-tags*) *theI*)
have $\forall\ z\ .\ [\boldsymbol{\mathcal{A}}\varphi\ z\ in\ v] = [(z^P) = (x^P)\ in\ v]$
proof
fix z
have $[\boldsymbol{\mathcal{A}}\varphi\ z\ in\ v] \Longrightarrow [(z^P) = (x^P)\ in\ v]$
unfolding *identity-κ-def* **apply** *meta-solver*
using *4 5 2* d_κ*-proper* w_0*-def* **by** *auto*
moreover have $[(z^P) = (x^P)\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi\ z\ in\ v]$
unfolding *identity-κ-def* **apply** *meta-solver*
using *2 4 5*
by (*simp add*: d_κ*-proper* w_0*-def*)
ultimately show $[\boldsymbol{\mathcal{A}}\varphi\ z\ in\ v] = [(z^P) = (x^P)\ in\ v]$
by *auto*
qed
thus $[\forall z.\ \boldsymbol{\mathcal{A}}\varphi\ z \equiv (z) = (x)\ in\ v]$
unfolding *identity-ν-def*
by (*simp add*: *AllI EquivS*)
next
fix v
assume $[\forall z.\ \boldsymbol{\mathcal{A}}\varphi\ z \equiv (z) = (x)\ in\ v]$
hence $\bigwedge z.\ (dw \models \varphi\ z) = (\exists\ o_1\ o_2.\ Some\ o_1 = d_\kappa\ (z^P)$
$\wedge\ Some\ o_2 = d_\kappa\ (x^P) \wedge o_1 = o_2)$
apply − **unfolding** *identity-ν-def identity-κ-def* **by** *meta-solver*
hence $\forall\ z\ .\ (dw \models \varphi\ z) = (z = x)$
by (*simp add*: d_κ*-proper*)
moreover hence $x = (THE\ z\ .\ (dw \models \varphi\ z))$ **by** *simp*
ultimately have $x^P = (\iota x.\ \varphi\ x)$
using *D3* d_κ*-inject* d_κ*-proper* w_0*-def* **by** *presburger*
thus $[x^P = (\iota x.\ \varphi\ x)\ in\ v]$
using *EqκS* **unfolding** *identity-κ-def* **by** (*metis* d_κ*-proper*)
qed

7.9 Axioms for Complex Relation Terms

lemma *lambda-predicates-1*[*axiom*]:
$(\boldsymbol{\lambda}\ x\ .\ \varphi\ x) = (\boldsymbol{\lambda}\ y\ .\ \varphi\ y)$ **..**

lemma *lambda-predicates-2-1*[*axiom*]:
assumes *IsProperInX* φ
shows $[[(\!|\boldsymbol{\lambda}\ x\ .\ \varphi\ (x^P),\ x^P|\!) \equiv \varphi\ (x^P)]]$
apply *axiom-meta-solver*
using *D5-1*[*OF assms*] d_κ*-proper* $propex_1$
by *metis*

lemma *lambda-predicates-2-2*[*axiom*]:
assumes *IsProperInXY* φ
shows $[[(\!|(\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \varphi\ (x^P)\ (y^P))),\ x^P,\ y^P|\!) \equiv \varphi\ (x^P)\ (y^P)]]$
apply *axiom-meta-solver*
using *D5-2*[*OF assms*] d_κ*-proper* $propex_2$
by *metis*

lemma *lambda-predicates-2-3* [*axiom*]:
assumes *IsProperInXYZ* φ
shows $[[\langle\!|(\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \varphi\ (x^P)\ (y^P)\ (z^P))),x^P,y^P,z^P|\!\rangle \equiv \varphi\ (x^P)\ (y^P)\ (z^P)]]$
proof −
have $[[\langle\!|(\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \varphi\ (x^P)\ (y^P)\ (z^P))),x^P,y^P,z^P|\!\rangle \to \varphi\ (x^P)\ (y^P)\ (z^P)]]$
apply *axiom-meta-solver* **using** *D5-3* [*OF assms*] **by** *auto*
moreover have
$[[\varphi\ (x^P)\ (y^P)\ (z^P) \to \langle\!|(\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \varphi\ (x^P)\ (y^P)\ (z^P))),x^P,y^P,z^P|\!\rangle]]$
apply *axiom-meta-solver*
using *D5-3* [*OF assms*] d_κ*-proper propex*$_3$
by (*metis* (*no-types*, *lifting*))
ultimately show *?thesis* **unfolding** *axiom-def equiv-def ConjS* **by** *blast*
qed

lemma *lambda-predicates-3-0* [*axiom*]:
$[[(\boldsymbol{\lambda}^0\ \varphi) = \varphi]]$
unfolding *identity-defs*
apply *axiom-meta-solver*
by (*simp add*: *meta-defs meta-aux*)

lemma *lambda-predicates-3-1* [*axiom*]:
$[[(\boldsymbol{\lambda}\ x\ .\ \langle\!|F,\ x^P|\!\rangle) = F]]$
unfolding *axiom-def*
apply (*rule allI*)
unfolding *identity*-Π_1-*def* **apply** (*rule* Eq_1I)
using *D4-1* d_1*-inject* **by** *simp*

lemma *lambda-predicates-3-2* [*axiom*]:
$[[(\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \langle\!|F,\ x^P,\ y^P|\!\rangle)) = F]]$
unfolding *axiom-def*
apply (*rule allI*)
unfolding *identity*-Π_2-*def* **apply** (*rule* Eq_2I)
using *D4-2* d_2*-inject* **by** *simp*

lemma *lambda-predicates-3-3* [*axiom*]:
$[[(\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \langle\!|F,\ x^P,\ y^P,\ z^P|\!\rangle)) = F]]$
unfolding *axiom-def*
apply (*rule allI*)
unfolding *identity*-Π_3-*def* **apply** (*rule* Eq_3I)
using *D4-3* d_3*-inject* **by** *simp*

lemma *lambda-predicates-4-0* [*axiom*]:
assumes $\bigwedge x.[(\boldsymbol{\mathcal{A}}(\varphi\ x \equiv \psi\ x))\ in\ v]$
shows $[[(\boldsymbol{\lambda}^0\ (\chi\ (\boldsymbol{\iota}x.\ \varphi\ x)) = \boldsymbol{\lambda}^0\ (\chi\ (\boldsymbol{\iota}x.\ \psi\ x)))]]$
unfolding *axiom-def identity*-o-*def* **apply** − **apply** (*rule allI*; *rule* Eq_0I)
using *TheEqI* [*OF assms* [*THEN ActualE*, *THEN EquivE*]] **by** *auto*

lemma *lambda-predicates-4-1* [*axiom*]:
assumes $\bigwedge x.[(\boldsymbol{\mathcal{A}}(\varphi\ x \equiv \psi\ x))\ in\ v]$
shows $[[((\boldsymbol{\lambda}\ x\ .\ \chi\ (\boldsymbol{\iota}x.\ \varphi\ x)\ x) = (\boldsymbol{\lambda}\ x\ .\ \chi\ (\boldsymbol{\iota}x.\ \psi\ x)\ x))]]$
unfolding *axiom-def identity*-Π_1-*def* **apply** − **apply** (*rule allI*; *rule* Eq_1I)
using *TheEqI* [*OF assms* [*THEN ActualE*, *THEN EquivE*]] **by** *auto*

lemma *lambda-predicates-4-2* [*axiom*]:
assumes $\bigwedge x.[(\boldsymbol{\mathcal{A}}(\varphi\ x \equiv \psi\ x))\ in\ v]$
shows $[[((\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \chi\ (\boldsymbol{\iota}x.\ \varphi\ x)\ x\ y)) = (\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \chi\ (\boldsymbol{\iota}x.\ \psi\ x)\ x\ y))))]]$
unfolding *axiom-def identity*-Π_2-*def* **apply** − **apply** (*rule allI*; *rule* Eq_2I)
using *TheEqI* [*OF assms* [*THEN ActualE*, *THEN EquivE*]] **by** *auto*

lemma *lambda-predicates-4-3* [*axiom*]:
assumes $\bigwedge x.[(\boldsymbol{\mathcal{A}}(\varphi\ x \equiv \psi\ x))\ in\ v]$
shows $[[(\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \chi\ (\boldsymbol{\iota}x.\ \varphi\ x)\ x\ y\ z)) = (\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \chi\ (\boldsymbol{\iota}x.\ \psi\ x)\ x\ y\ z))]]$
unfolding *axiom-def identity*-Π_3-*def* **apply** − **apply** (*rule allI*; *rule* Eq_3I)

using *TheEqI*[*OF assms*[*THEN ActualE*, *THEN EquivE*]] **by** *auto*

7.10 Axioms of Encoding

lemma *encoding*[*axiom*]:
[[$\{\!|x,F|\!\} \rightarrow \Box\{\!|x,F|\!\}$]]
by *axiom-meta-solver*
lemma *nocoder*[*axiom*]:
[[$(\!|O!,x|\!) \rightarrow \neg(\exists\ F\ .\ \{\!|x,F|\!\})$]]
unfolding *axiom-def*
apply (*rule allI*, *rule ImplI*, *subst* (*asm*) *OrdS*)
apply *meta-solver* **unfolding** *en-def*
by (*metis* ν*.simps*(*5*) *mem-Collect-eq option.sel*)
lemma *A-objects*[*axiom*]:
[[$\exists\,x.\ (\!|A!,x^P|\!)$ & $(\forall\ F\ .\ (\{\!|x^P,F|\!\} \equiv \varphi\ F))$]]
unfolding *axiom-def*
proof (*rule allI*, *rule ExIRule*)
fix *v*
let *?x* = $\alpha\nu$ { *F* . [φ *F in v*]}
have [$(\!|A!,?x^P|\!)$ *in v*] **by** (*simp add*: *AbsS* d_κ*-proper*)
moreover have [$(\forall\,F.\ \{\!|?x^P,F|\!\} \equiv \varphi\ F)$ *in v*]
apply *meta-solver* **unfolding** *en-def*
using d_1*.rep-eq* d_κ*-def* d_κ*-proper eval*Π_1*-inverse* **by** *auto*
ultimately show [$(\!|A!,?x^P|\!)$ & $(\forall\,F.\ \{\!|?x^P,F|\!\} \equiv \varphi\ F)$ *in v*]
by (*simp only*: *ConjS*)
qed

end

8 Definitions

8.1 Property Negations

consts *propnot* :: $'a\Rightarrow'a$ (-$^-$ [*90*] *90*)
overloading $propnot_0 \equiv propnot :: \Pi_0\Rightarrow\Pi_0$
$propnot_1 \equiv propnot :: \Pi_1\Rightarrow\Pi_1$
$propnot_2 \equiv propnot :: \Pi_2\Rightarrow\Pi_2$
$propnot_3 \equiv propnot :: \Pi_3\Rightarrow\Pi_3$
begin
definition $propnot_0 :: \Pi_0\Rightarrow\Pi_0$ **where**
$propnot_0 \equiv \lambda\ p\ .\ \boldsymbol{\lambda}^0\ (\neg p)$
definition $propnot_1$ **where**
$propnot_1 \equiv \lambda\ F\ .\ \boldsymbol{\lambda}\ x\ .\ \neg(\!|F,\ x^P|\!)$
definition $propnot_2$ **where**
$propnot_2 \equiv \lambda\ F\ .\ \boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \neg(\!|F,\ x^P,\ y^P|\!))$
definition $propnot_3$ **where**
$propnot_3 \equiv \lambda\ F\ .\ \boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \neg(\!|F,\ x^P,\ y^P,\ z^P|\!))$
end

named-theorems *propnot-defs*
declare $propnot_0$*-def*[*propnot-defs*] $propnot_1$*-def*[*propnot-defs*]
$propnot_2$*-def*[*propnot-defs*] $propnot_3$*-def*[*propnot-defs*]

8.2 Noncontingent and Contingent Relations

consts *Necessary* :: $'a\Rightarrow o$
overloading $Necessary_0 \equiv Necessary :: \Pi_0\Rightarrow o$
$Necessary_1 \equiv Necessary :: \Pi_1\Rightarrow o$
$Necessary_2 \equiv Necessary :: \Pi_2\Rightarrow o$
$Necessary_3 \equiv Necessary :: \Pi_3\Rightarrow o$
begin

definition $\mathit{Necessary}_0$ **where**
$\mathit{Necessary}_0 \equiv \lambda\ p\ .\ \Box p$
definition $\mathit{Necessary}_1 :: \Pi_1 \Rightarrow o$ **where**
$\mathit{Necessary}_1 \equiv \lambda\ F\ .\ \Box(\forall\ \ x\ .\ (\!|F,x^P|\!))$
definition $\mathit{Necessary}_2$ **where**
$\mathit{Necessary}_2 \equiv \lambda\ F\ .\ \Box(\forall\ \ x\ y\ .\ (\!|F,x^P,y^P|\!))$
definition $\mathit{Necessary}_3$ **where**
$\mathit{Necessary}_3 \equiv \lambda\ F\ .\ \Box(\forall\ \ x\ y\ z\ .\ (\!|F,x^P,y^P,z^P|\!))$
end

named-theorems *Necessary-defs*
declare $\mathit{Necessary}_0$*-def*[*Necessary-defs*] $\mathit{Necessary}_1$*-def*[*Necessary-defs*]
$\mathit{Necessary}_2$*-def*[*Necessary-defs*] $\mathit{Necessary}_3$*-def*[*Necessary-defs*]

consts $\mathit{Impossible} :: {'a} \Rightarrow o$
overloading $\mathit{Impossible}_0 \equiv \mathit{Impossible} :: \Pi_0 \Rightarrow o$
$\mathit{Impossible}_1 \equiv \mathit{Impossible} :: \Pi_1 \Rightarrow o$
$\mathit{Impossible}_2 \equiv \mathit{Impossible} :: \Pi_2 \Rightarrow o$
$\mathit{Impossible}_3 \equiv \mathit{Impossible} :: \Pi_3 \Rightarrow o$
begin
definition $\mathit{Impossible}_0$ **where**
$\mathit{Impossible}_0 \equiv \lambda\ p\ .\ \Box\neg p$
definition $\mathit{Impossible}_1$ **where**
$\mathit{Impossible}_1 \equiv \lambda\ F\ .\ \Box(\forall\ \ x.\ \neg(\!|F,x^P|\!))$
definition $\mathit{Impossible}_2$ **where**
$\mathit{Impossible}_2 \equiv \lambda\ F\ .\ \Box(\forall\ \ x\ y.\ \neg(\!|F,x^P,y^P|\!))$
definition $\mathit{Impossible}_3$ **where**
$\mathit{Impossible}_3 \equiv \lambda\ F\ .\ \Box(\forall\ \ x\ y\ z.\ \neg(\!|F,x^P,y^P,z^P|\!))$
end

named-theorems *Impossible-defs*
declare $\mathit{Impossible}_0$*-def*[*Impossible-defs*] $\mathit{Impossible}_1$*-def*[*Impossible-defs*]
$\mathit{Impossible}_2$*-def*[*Impossible-defs*] $\mathit{Impossible}_3$*-def*[*Impossible-defs*]

definition *NonContingent* **where**
$\mathit{NonContingent} \equiv \lambda\ F\ .\ (\mathit{Necessary}\ F) \vee (\mathit{Impossible}\ F)$
definition *Contingent* **where**
$\mathit{Contingent} \equiv \lambda\ F\ .\ \ \neg(\mathit{Necessary}\ F \vee \mathit{Impossible}\ F)$

definition *ContingentlyTrue* $:: o \Rightarrow o$ **where**
$\mathit{ContingentlyTrue} \equiv \lambda\ p\ .\ p\ \&\ \Diamond\neg p$
definition *ContingentlyFalse* $:: o \Rightarrow o$ **where**
$\mathit{ContingentlyFalse} \equiv \lambda\ p\ .\ \neg p\ \&\ \Diamond p$

definition *WeaklyContingent* **where**
$\mathit{WeaklyContingent} \equiv \lambda\ F\ .\ \mathit{Contingent}\ F\ \&\ (\forall\ \ x.\ \Diamond(\!|F,x^P|\!) \longrightarrow \Box(\!|F,x^P|\!))$

8.3 Null and Universal Objects

definition *Null* $:: \kappa \Rightarrow o$ **where**
$\mathit{Null} \equiv \lambda\ x\ .\ (\!|A!,x|\!)\ \&\ \neg(\exists\ \ F\ .\ \{\!|x,\ F|\!\})$
definition *Universal* $:: \kappa \Rightarrow o$ **where**
$\mathit{Universal} \equiv \lambda\ x\ .\ (\!|A!,x|\!)\ \&\ (\forall\ \ F\ .\ \{\!|x,\ F|\!\})$

definition *NullObject* $:: \kappa\ (\mathbf{a}_\emptyset)$ **where**
$\mathit{NullObject} \equiv (\iota x\ .\ \mathit{Null}\ (x^P))$
definition *UniversalObject* $:: \kappa\ (\mathbf{a}_V)$ **where**
$\mathit{UniversalObject} \equiv (\iota x\ .\ \mathit{Universal}\ (x^P))$

8.4 Propositional Properties

definition *Propositional* **where**
$\mathit{Propositional}\ F \equiv \exists\ \ p\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)$

8.5 Indiscriminate Properties

definition *Indiscriminate* :: $\Pi_1 \Rightarrow o$ **where**
Indiscriminate $\equiv \lambda\ F\ .\ \Box((\exists\ \ x\ .\ (|F,x^P|)) \rightarrow (\forall\ \ x\ .\ (|F,x^P|)))$

8.6 Miscellaneous

definition *not-identical*$_E$ :: $\kappa \Rightarrow \kappa \Rightarrow o$ (**infixl** $\neq_E$ *63*)
where *not-identical*$_E \equiv \lambda\ x\ y\ .\ (|(\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ x^P\ =_E\ y^P))^-,\ x,\ y|)$

9 The Deductive System PLM

declare *meta-defs*[*no-atp*] *meta-aux*[*no-atp*]

locale *PLM* = *Axioms*
begin

9.1 Automatic Solver

named-theorems *PLM*
named-theorems *PLM-intro*
named-theorems *PLM-elim*
named-theorems *PLM-dest*
named-theorems *PLM-subst*

method *PLM-solver* **declares** *PLM-intro PLM-elim PLM-subst PLM-dest PLM*
= ((*assumption* | (*match axiom* **in** *A*: $[[\varphi]]$ **for** $\varphi \Rightarrow$ ‹*fact A*[*axiom-instance*]›)
| *fact PLM* | *rule PLM-intro* | *subst PLM-subst* | *subst* (*asm*) *PLM-subst*
| *fastforce* | *safe* | *drule PLM-dest* | *erule PLM-elim*); (*PLM-solver*)*?*)

9.2 Modus Ponens

lemma *modus-ponens*[*PLM*]:
$[\![[\varphi\ in\ v];\ [\varphi \rightarrow \psi\ in\ v]]\!] \Longrightarrow [\psi\ in\ v]$
by (*simp add*: *Semantics.T5*)

9.3 Axioms

interpretation *Axioms* **.**
declare *axiom*[*PLM*]
declare *conn-defs*[*PLM*]

9.4 (Modally Strict) Proofs and Derivations

lemma *vdash-properties-6*[*no-atp*]:
$[\![[\varphi\ in\ v];\ [\varphi \rightarrow \psi\ in\ v]]\!] \Longrightarrow [\psi\ in\ v]$
using *modus-ponens* **.**
lemma *vdash-properties-9*[*PLM*]:
$[\varphi\ in\ v] \Longrightarrow [\psi \rightarrow \varphi\ in\ v]$
using *modus-ponens pl-1*[*axiom-instance*] **by** *blast*
lemma *vdash-properties-10*[*PLM*]:
$[\varphi \rightarrow \psi\ in\ v] \Longrightarrow ([\varphi\ in\ v] \Longrightarrow [\psi\ in\ v])$
using *vdash-properties-6* **.**

attribute-setup *deduction* = ⟪
Scan.succeed (*Thm.rule-attribute* []
(*fn* - => *fn thm* => *thm RS* @{*thm vdash-properties-10*}))
⟫

9.5 GEN and RN

lemma *rule-gen*[*PLM*]:
$[\![\bigwedge\alpha\ .\ [\varphi\ \alpha\ in\ v]]\!] \Longrightarrow [\forall\,\alpha\ .\ \varphi\ \alpha\ in\ v]$
by (*simp add*: *Semantics.T8*)

lemma *RN-2*[*PLM*]:
$(\bigwedge\ v\ .\ [\psi\ in\ v] \Longrightarrow [\varphi\ in\ v]) \Longrightarrow ([\Box\psi\ in\ v] \Longrightarrow [\Box\varphi\ in\ v])$
by (*simp add*: *Semantics.T6*)

lemma *RN*[*PLM*]:
$(\bigwedge\ v\ .\ [\varphi\ in\ v]) \Longrightarrow [\Box\varphi\ in\ v]$
using *qml-3*[*axiom-necessitation*, *axiom-instance*] *RN-2* **by** *blast*

9.6 Negations and Conditionals

lemma *if-p-then-p*[*PLM*]:
$[\varphi \rightarrow \varphi\ in\ v]$
using *pl-1 pl-2 vdash-properties-10 axiom-instance* **by** *blast*

lemma *deduction-theorem*[*PLM*,*PLM-intro*]:
$[\![[\varphi\ in\ v] \Longrightarrow [\psi\ in\ v]]\!] \Longrightarrow [\varphi \rightarrow \psi\ in\ v]$
by (*simp add*: *Semantics.T5*)
lemmas *CP* = *deduction-theorem*

lemma *ded-thm-cor-3*[*PLM*]:
$[\![[\varphi \rightarrow \psi\ in\ v];\ [\psi \rightarrow \chi\ in\ v]]\!] \Longrightarrow [\varphi \rightarrow \chi\ in\ v]$
by (*meson pl-2 vdash-properties-10 vdash-properties-9 axiom-instance*)
lemma *ded-thm-cor-4*[*PLM*]:
$[\![[\varphi \rightarrow (\psi \rightarrow \chi)\ in\ v];\ [\psi\ in\ v]]\!] \Longrightarrow [\varphi \rightarrow \chi\ in\ v]$
by (*meson pl-2 vdash-properties-10 vdash-properties-9 axiom-instance*)

lemma *useful-tautologies-1*[*PLM*]:
$[\neg\neg\varphi \rightarrow \varphi\ in\ v]$
by (*meson pl-1 pl-3 ded-thm-cor-3 ded-thm-cor-4 axiom-instance*)
lemma *useful-tautologies-2*[*PLM*]:
$[\varphi \rightarrow \neg\neg\varphi\ in\ v]$
by (*meson pl-1 pl-3 ded-thm-cor-3 useful-tautologies-1*
vdash-properties-10 axiom-instance)
lemma *useful-tautologies-3*[*PLM*]:
$[\neg\varphi \rightarrow (\varphi \rightarrow \psi)\ in\ v]$
by (*meson pl-1 pl-2 pl-3 ded-thm-cor-3 ded-thm-cor-4 axiom-instance*)
lemma *useful-tautologies-4*[*PLM*]:
$[(\neg\psi \rightarrow \neg\varphi) \rightarrow (\varphi \rightarrow \psi)\ in\ v]$
by (*meson pl-1 pl-2 pl-3 ded-thm-cor-3 ded-thm-cor-4 axiom-instance*)
lemma *useful-tautologies-5*[*PLM*]:
$[(\varphi \rightarrow \psi) \rightarrow (\neg\psi \rightarrow \neg\varphi)\ in\ v]$
by (*metis CP useful-tautologies-4 vdash-properties-10*)
lemma *useful-tautologies-6*[*PLM*]:
$[(\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \neg\varphi)\ in\ v]$
by (*metis CP useful-tautologies-4 vdash-properties-10*)
lemma *useful-tautologies-7*[*PLM*]:
$[(\neg\varphi \rightarrow \psi) \rightarrow (\neg\psi \rightarrow \varphi)\ in\ v]$
using *ded-thm-cor-3 useful-tautologies-4 useful-tautologies-5*
useful-tautologies-6 **by** *blast*
lemma *useful-tautologies-8*[*PLM*]:
$[\varphi \rightarrow (\neg\psi \rightarrow \neg(\varphi \rightarrow \psi))\ in\ v]$
by (*meson ded-thm-cor-3 CP useful-tautologies-5*)
lemma *useful-tautologies-9*[*PLM*]:
$[(\varphi \rightarrow \psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \psi)\ in\ v]$
by (*metis CP useful-tautologies-4 vdash-properties-10*)
lemma *useful-tautologies-10*[*PLM*]:
$[(\varphi \rightarrow \neg\psi) \rightarrow ((\varphi \rightarrow \psi) \rightarrow \neg\varphi)\ in\ v]$

by (*metis ded-thm-cor-3 CP useful-tautologies-6*)

lemma *modus-tollens-1*[*PLM*]:
$[\![[\varphi \rightarrow \psi \ in \ v]; [\neg\psi \ in \ v]]\!] \Longrightarrow [\neg\varphi \ in \ v]$
by (*metis ded-thm-cor-3 ded-thm-cor-4 useful-tautologies-3 useful-tautologies-7 vdash-properties-10*)
lemma *modus-tollens-2*[*PLM*]:
$[\![[\varphi \rightarrow \neg\psi \ in \ v]; [\psi \ in \ v]]\!] \Longrightarrow [\neg\varphi \ in \ v]$
using *modus-tollens-1 useful-tautologies-2 vdash-properties-10* **by** *blast*

lemma *contraposition-1*[*PLM*]:
$[\varphi \rightarrow \psi \ in \ v] = [\neg\psi \rightarrow \neg\varphi \ in \ v]$
using *useful-tautologies-4 useful-tautologies-5 vdash-properties-10* **by** *blast*
lemma *contraposition-2*[*PLM*]:
$[\varphi \rightarrow \neg\psi \ in \ v] = [\psi \rightarrow \neg\varphi \ in \ v]$
using *contraposition-1 ded-thm-cor-3 useful-tautologies-1* **by** *blast*

lemma *reductio-aa-1*[*PLM*]:
$[\![[\neg\varphi \ in \ v] \Longrightarrow [\neg\psi \ in \ v]; [\neg\varphi \ in \ v] \Longrightarrow [\psi \ in \ v]]\!] \Longrightarrow [\varphi \ in \ v]$
using *CP modus-tollens-2 useful-tautologies-1 vdash-properties-10* **by** *blast*
lemma *reductio-aa-2*[*PLM*]:
$[\![[\varphi \ in \ v] \Longrightarrow [\neg\psi \ in \ v]; [\varphi \ in \ v] \Longrightarrow [\psi \ in \ v]]\!] \Longrightarrow [\neg\varphi \ in \ v]$
by (*meson contraposition-1 reductio-aa-1*)
lemma *reductio-aa-3*[*PLM*]:
$[\![[\neg\varphi \rightarrow \neg\psi \ in \ v]; [\neg\varphi \rightarrow \psi \ in \ v]]\!] \Longrightarrow [\varphi \ in \ v]$
using *reductio-aa-1 vdash-properties-10* **by** *blast*
lemma *reductio-aa-4*[*PLM*]:
$[\![[\varphi \rightarrow \neg\psi \ in \ v]; [\varphi \rightarrow \psi \ in \ v]]\!] \Longrightarrow [\neg\varphi \ in \ v]$
using *reductio-aa-2 vdash-properties-10* **by** *blast*

lemma *raa-cor-1*[*PLM*]:
$[\![[\varphi \ in \ v]; [\neg\psi \ in \ v] \Longrightarrow [\neg\varphi \ in \ v]]\!] \Longrightarrow ([\varphi \ in \ v] \Longrightarrow [\psi \ in \ v])$
using *reductio-aa-1 vdash-properties-9* **by** *blast*
lemma *raa-cor-2*[*PLM*]:
$[\![[\neg\varphi \ in \ v]; [\neg\psi \ in \ v] \Longrightarrow [\varphi \ in \ v]]\!] \Longrightarrow ([\neg\varphi \ in \ v] \Longrightarrow [\psi \ in \ v])$
using *reductio-aa-1 vdash-properties-9* **by** *blast*
lemma *raa-cor-3*[*PLM*]:
$[\![[\varphi \ in \ v]; [\neg\psi \rightarrow \neg\varphi \ in \ v]]\!] \Longrightarrow ([\varphi \ in \ v] \Longrightarrow [\psi \ in \ v])$
using *raa-cor-1 vdash-properties-10* **by** *blast*
lemma *raa-cor-4*[*PLM*]:
$[\![[\neg\varphi \ in \ v]; [\neg\psi \rightarrow \varphi \ in \ v]]\!] \Longrightarrow ([\neg\varphi \ in \ v] \Longrightarrow [\psi \ in \ v])$
using *raa-cor-2 vdash-properties-10* **by** *blast*

Remark 20. *In contrast to PLM the classical introduction and elimination rules are proven before the tautologies. The statements proven so far are sufficient for the proofs and using the derived rules the tautologies can be derived automatically.*

lemma *intro-elim-1*[*PLM*]:
$[\![[\varphi \ in \ v]; [\psi \ in \ v]]\!] \Longrightarrow [\varphi \ \& \ \psi \ in \ v]$
unfolding *conj-def* **using** *ded-thm-cor-4 if-p-then-p modus-tollens-2* **by** *blast*
lemmas **&***I* = *intro-elim-1*
lemma *intro-elim-2-a*[*PLM*]:
$[\varphi \ \& \ \psi \ in \ v] \Longrightarrow [\varphi \ in \ v]$
unfolding *conj-def* **using** *CP reductio-aa-1* **by** *blast*
lemma *intro-elim-2-b*[*PLM*]:
$[\varphi \ \& \ \psi \ in \ v] \Longrightarrow [\psi \ in \ v]$
unfolding *conj-def* **using** *pl-1 CP reductio-aa-1 axiom-instance* **by** *blast*
lemmas **&***E* = *intro-elim-2-a intro-elim-2-b*
lemma *intro-elim-3-a*[*PLM*]:
$[\varphi \ in \ v] \Longrightarrow [\varphi \lor \psi \ in \ v]$

unfolding *disj-def* **using** *ded-thm-cor-4 useful-tautologies-3* **by** *blast*
lemma *intro-elim-3-b*[*PLM*]:
$[\psi\ in\ v] \Longrightarrow [\varphi \lor \psi\ in\ v]$
by (*simp only*: *disj-def vdash-properties-9*)
lemmas $\lor I$ = *intro-elim-3-a intro-elim-3-b*
lemma *intro-elim-4-a*[*PLM*]:
$[\![[\varphi \lor \psi\ in\ v]; [\varphi \rightarrow \chi\ in\ v]; [\psi \rightarrow \chi\ in\ v]]\!] \Longrightarrow [\chi\ in\ v]$
unfolding *disj-def* **by** (*meson reductio-aa-2 vdash-properties-10*)
lemma *intro-elim-4-b*[*PLM*]:
$[\![[\varphi \lor \psi\ in\ v]; [\neg\varphi\ in\ v]]\!] \Longrightarrow [\psi\ in\ v]$
unfolding *disj-def* **using** *vdash-properties-10* **by** *blast*
lemma *intro-elim-4-c*[*PLM*]:
$[\![[\varphi \lor \psi\ in\ v]; [\neg\psi\ in\ v]]\!] \Longrightarrow [\varphi\ in\ v]$
unfolding *disj-def* **using** *raa-cor-2 vdash-properties-10* **by** *blast*
lemma *intro-elim-4-d*[*PLM*]:
$[\![[\varphi \lor \psi\ in\ v]; [\varphi \rightarrow \chi\ in\ v]; [\psi \rightarrow \Theta\ in\ v]]\!] \Longrightarrow [\chi \lor \Theta\ in\ v]$
unfolding *disj-def* **using** *contraposition-1 ded-thm-cor-3* **by** *blast*
lemma *intro-elim-4-e*[*PLM*]:
$[\![[\varphi \lor \psi\ in\ v]; [\varphi \equiv \chi\ in\ v]; [\psi \equiv \Theta\ in\ v]]\!] \Longrightarrow [\chi \lor \Theta\ in\ v]$
unfolding *equiv-def* **using** &E(*1*) *intro-elim-4-d* **by** *blast*
lemmas $\lor E$ = *intro-elim-4-a intro-elim-4-b intro-elim-4-c intro-elim-4-d*
lemma *intro-elim-5*[*PLM*]:
$[\![[\varphi \rightarrow \psi\ in\ v]; [\psi \rightarrow \varphi\ in\ v]]\!] \Longrightarrow [\varphi \equiv \psi\ in\ v]$
by (*simp only*: *equiv-def* &I)
lemmas $\equiv I$ = *intro-elim-5*
lemma *intro-elim-6-a*[*PLM*]:
$[\![[\varphi \equiv \psi\ in\ v]; [\varphi\ in\ v]]\!] \Longrightarrow [\psi\ in\ v]$
unfolding *equiv-def* **using** &E(*1*) *vdash-properties-10* **by** *blast*
lemma *intro-elim-6-b*[*PLM*]:
$[\![[\varphi \equiv \psi\ in\ v]; [\psi\ in\ v]]\!] \Longrightarrow [\varphi\ in\ v]$
unfolding *equiv-def* **using** &E(*2*) *vdash-properties-10* **by** *blast*
lemma *intro-elim-6-c*[*PLM*]:
$[\![[\varphi \equiv \psi\ in\ v]; [\neg\varphi\ in\ v]]\!] \Longrightarrow [\neg\psi\ in\ v]$
unfolding *equiv-def* **using** &E(*2*) *modus-tollens-1* **by** *blast*
lemma *intro-elim-6-d*[*PLM*]:
$[\![[\varphi \equiv \psi\ in\ v]; [\neg\psi\ in\ v]]\!] \Longrightarrow [\neg\varphi\ in\ v]$
unfolding *equiv-def* **using** &E(*1*) *modus-tollens-1* **by** *blast*
lemma *intro-elim-6-e*[*PLM*]:
$[\![[\varphi \equiv \psi\ in\ v]; [\psi \equiv \chi\ in\ v]]\!] \Longrightarrow [\varphi \equiv \chi\ in\ v]$
by (*metis equiv-def ded-thm-cor-3* &E $\equiv I$)
lemma *intro-elim-6-f*[*PLM*]:
$[\![[\varphi \equiv \psi\ in\ v]; [\varphi \equiv \chi\ in\ v]]\!] \Longrightarrow [\chi \equiv \psi\ in\ v]$
by (*metis equiv-def ded-thm-cor-3* &E $\equiv I$)
lemmas $\equiv E$ = *intro-elim-6-a intro-elim-6-b intro-elim-6-c*
intro-elim-6-d intro-elim-6-e intro-elim-6-f
lemma *intro-elim-7*[*PLM*]:
$[\varphi\ in\ v] \Longrightarrow [\neg\neg\varphi\ in\ v]$
using *if-p-then-p modus-tollens-2* **by** *blast*
lemmas $\neg\neg I$ = *intro-elim-7*
lemma *intro-elim-8*[*PLM*]:
$[\neg\neg\varphi\ in\ v] \Longrightarrow [\varphi\ in\ v]$
using *if-p-then-p raa-cor-2* **by** *blast*
lemmas $\neg\neg E$ = *intro-elim-8*

context
begin
private lemma *NotNotI*[*PLM-intro*]:
$[\varphi\ in\ v] \Longrightarrow [\neg(\neg\varphi)\ in\ v]$
by (*simp add*: $\neg\neg I$)
private lemma *NotNotD*[*PLM-dest*]:
$[\neg(\neg\varphi)\ in\ v] \Longrightarrow [\varphi\ in\ v]$
using $\neg\neg E$ **by** *blast*

private lemma *ImplI*[*PLM-intro*]:
$([\varphi\ in\ v] \Longrightarrow [\psi\ in\ v]) \Longrightarrow [\varphi \rightarrow \psi\ in\ v]$
using *CP* **.**
private lemma *ImplE*[*PLM-elim*, *PLM-dest*]:
$[\varphi \rightarrow \psi\ in\ v] \Longrightarrow ([\varphi\ in\ v] \Longrightarrow [\psi\ in\ v])$
using *modus-ponens* **.**
private lemma *ImplS*[*PLM-subst*]:
$[\varphi \rightarrow \psi\ in\ v] = ([\varphi\ in\ v] \longrightarrow [\psi\ in\ v])$
using *ImplI ImplE* **by** *blast*

private lemma *NotI*[*PLM-intro*]:
$([\varphi\ in\ v] \Longrightarrow (\bigwedge\psi\ .[\psi\ in\ v])) \Longrightarrow [\neg\varphi\ in\ v]$
using *CP modus-tollens-2* **by** *blast*
private lemma *NotE*[*PLM-elim*,*PLM-dest*]:
$[\neg\varphi\ in\ v] \Longrightarrow ([\varphi\ in\ v] \longrightarrow (\forall\psi\ .[\psi\ in\ v]))$
using $\vee I(2)$ $\vee E(3)$ **by** *blast*
private lemma *NotS*[*PLM-subst*]:
$[\neg\varphi\ in\ v] = ([\varphi\ in\ v] \longrightarrow (\forall\psi\ .[\psi\ in\ v]))$
using *NotI NotE* **by** *blast*

private lemma *ConjI*[*PLM-intro*]:
$[\![[\varphi\ in\ v];\ [\psi\ in\ v]]\!] \Longrightarrow [\varphi\ \&\ \psi\ in\ v]$
using $\&I$ **by** *blast*
private lemma *ConjE*[*PLM-elim*,*PLM-dest*]:
$[\varphi\ \&\ \psi\ in\ v] \Longrightarrow (([\varphi\ in\ v] \wedge [\psi\ in\ v]))$
using *CP* $\&E$ **by** *blast*
private lemma *ConjS*[*PLM-subst*]:
$[\varphi\ \&\ \psi\ in\ v] = (([\varphi\ in\ v] \wedge [\psi\ in\ v]))$
using *ConjI ConjE* **by** *blast*

private lemma *DisjI*[*PLM-intro*]:
$[\varphi\ in\ v] \vee [\psi\ in\ v] \Longrightarrow [\varphi \vee \psi\ in\ v]$
using $\vee I$ **by** *blast*
private lemma *DisjE*[*PLM-elim*,*PLM-dest*]:
$[\varphi \vee \psi\ in\ v] \Longrightarrow [\varphi\ in\ v] \vee [\psi\ in\ v]$
using *CP* $\vee E(1)$ **by** *blast*
private lemma *DisjS*[*PLM-subst*]:
$[\varphi \vee \psi\ in\ v] = ([\varphi\ in\ v] \vee [\psi\ in\ v])$
using *DisjI DisjE* **by** *blast*

private lemma *EquivI*[*PLM-intro*]:
$[\![[\varphi\ in\ v] \Longrightarrow [\psi\ in\ v];[\psi\ in\ v] \Longrightarrow [\varphi\ in\ v]]\!] \Longrightarrow [\varphi \equiv \psi\ in\ v]$
using *CP* $\equiv I$ **by** *blast*
private lemma *EquivE*[*PLM-elim*,*PLM-dest*]:
$[\varphi \equiv \psi\ in\ v] \Longrightarrow ((([\varphi\ in\ v] \longrightarrow [\psi\ in\ v]) \wedge ([\psi\ in\ v] \longrightarrow [\varphi\ in\ v]))$
using $\equiv E(1)$ $\equiv E(2)$ **by** *blast*
private lemma *EquivS*[*PLM-subst*]:
$[\varphi \equiv \psi\ in\ v] = ([\varphi\ in\ v] \longleftrightarrow [\psi\ in\ v])$
using *EquivI EquivE* **by** *blast*

private lemma *NotOrD*[*PLM-dest*]:
$\neg[\varphi \vee \psi\ in\ v] \Longrightarrow \neg[\varphi\ in\ v] \wedge \neg[\psi\ in\ v]$
using $\vee I$ **by** *blast*
private lemma *NotAndD*[*PLM-dest*]:
$\neg[\varphi\ \&\ \psi\ in\ v] \Longrightarrow \neg[\varphi\ in\ v] \vee \neg[\psi\ in\ v]$
using $\&I$ **by** *blast*
private lemma *NotEquivD*[*PLM-dest*]:
$\neg[\varphi \equiv \psi\ in\ v] \Longrightarrow [\varphi\ in\ v] \neq [\psi\ in\ v]$
by (*meson NotI contraposition-1* $\equiv I$ *vdash-properties-9*)

private lemma *BoxI*[*PLM-intro*]:
$(\bigwedge\ v\ .\ [\varphi\ in\ v]) \Longrightarrow [\Box\varphi\ in\ v]$
using *RN* **by** *blast*

private lemma *NotBoxD*[*PLM-dest*]:
$\neg[\Box\varphi\ in\ v] \Longrightarrow (\exists\ v\ .\ \neg[\varphi\ in\ v])$
using *BoxI* **by** *blast*

private lemma *AllI*[*PLM-intro*]:
$(\bigwedge\ x\ .\ [\varphi\ x\ in\ v]) \Longrightarrow [\forall\ x\ .\ \varphi\ x\ in\ v]$
using *rule-gen* **by** *blast*
lemma *NotAllD*[*PLM-dest*]:
$\neg[\forall\ x\ .\ \varphi\ x\ in\ v] \Longrightarrow (\exists\ x\ .\ \neg[\varphi\ x\ in\ v])$
using *AllI* **by** *fastforce*
end

lemma *oth-class-taut-1-a*[*PLM*]:
$[\neg(\varphi\ \&\ \neg\varphi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-1-b*[*PLM*]:
$[\neg(\varphi \equiv \neg\varphi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-2*[*PLM*]:
$[\varphi \lor \neg\varphi\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-a*[*PLM*]:
$[(\varphi\ \&\ \varphi) \equiv \varphi\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-b*[*PLM*]:
$[(\varphi\ \&\ \psi) \equiv (\psi\ \&\ \varphi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-c*[*PLM*]:
$[(\varphi\ \&\ (\psi\ \&\ \chi)) \equiv ((\varphi\ \&\ \psi)\ \&\ \chi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-d*[*PLM*]:
$[(\varphi \lor \varphi) \equiv \varphi\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-e*[*PLM*]:
$[(\varphi \lor \psi) \equiv (\psi \lor \varphi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-f*[*PLM*]:
$[(\varphi \lor (\psi \lor \chi)) \equiv ((\varphi \lor \psi) \lor \chi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-g*[*PLM*]:
$[(\varphi \equiv \psi) \equiv (\psi \equiv \varphi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-i*[*PLM*]:
$[(\varphi \equiv (\psi \equiv \chi)) \equiv ((\varphi \equiv \psi) \equiv \chi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-4-a*[*PLM*]:
$[\varphi \equiv \varphi\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-4-b*[*PLM*]:
$[\varphi \equiv \neg\neg\varphi\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-5-a*[*PLM*]:
$[(\varphi \rightarrow \psi) \equiv \neg(\varphi\ \&\ \neg\psi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-5-b*[*PLM*]:
$[\neg(\varphi \rightarrow \psi) \equiv (\varphi\ \&\ \neg\psi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-5-c*[*PLM*]:
$[(\varphi \rightarrow \psi) \rightarrow ((\psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \chi))\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-5-d*[*PLM*]:
$[(\varphi \equiv \psi) \equiv (\neg\varphi \equiv \neg\psi)\ in\ v]$
by *PLM-solver*

lemma *oth-class-taut-5-e*[*PLM*]:
  $[(\varphi \equiv \psi) \rightarrow ((\varphi \rightarrow \chi) \equiv (\psi \rightarrow \chi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-5-f*[*PLM*]:
  $[(\varphi \equiv \psi) \rightarrow ((\chi \rightarrow \varphi) \equiv (\chi \rightarrow \psi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-5-g*[*PLM*]:
  $[(\varphi \equiv \psi) \rightarrow ((\varphi \equiv \chi) \equiv (\psi \equiv \chi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-5-h*[*PLM*]:
  $[(\varphi \equiv \psi) \rightarrow ((\chi \equiv \varphi) \equiv (\chi \equiv \psi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-5-i*[*PLM*]:
  $[(\varphi \equiv \psi) \equiv ((\varphi\ \&\ \psi) \vee (\neg\varphi\ \&\ \neg\psi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-5-j*[*PLM*]:
  $[(\neg(\varphi \equiv \psi)) \equiv ((\varphi\ \&\ \neg\psi) \vee (\neg\varphi\ \&\ \psi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-5-k*[*PLM*]:
  $[(\varphi \rightarrow \psi) \equiv (\neg\varphi \vee \psi)\ \mathit{in}\ v]$
  by *PLM-solver*

lemma *oth-class-taut-6-a*[*PLM*]:
  $[(\varphi\ \&\ \psi) \equiv \neg(\neg\varphi \vee \neg\psi)\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-6-b*[*PLM*]:
  $[(\varphi \vee \psi) \equiv \neg(\neg\varphi\ \&\ \neg\psi)\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-6-c*[*PLM*]:
  $[\neg(\varphi\ \&\ \psi) \equiv (\neg\varphi \vee \neg\psi)\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-6-d*[*PLM*]:
  $[\neg(\varphi \vee \psi) \equiv (\neg\varphi\ \&\ \neg\psi)\ \mathit{in}\ v]$
  by *PLM-solver*

lemma *oth-class-taut-7-a*[*PLM*]:
  $[(\varphi\ \&\ (\psi \vee \chi)) \equiv ((\varphi\ \&\ \psi) \vee (\varphi\ \&\ \chi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-7-b*[*PLM*]:
  $[(\varphi \vee (\psi\ \&\ \chi)) \equiv ((\varphi \vee \psi)\ \&\ (\varphi \vee \chi))\ \mathit{in}\ v]$
  by *PLM-solver*

lemma *oth-class-taut-8-a*[*PLM*]:
  $[((\varphi\ \&\ \psi) \rightarrow \chi) \rightarrow (\varphi \rightarrow (\psi \rightarrow \chi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-8-b*[*PLM*]:
  $[(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi\ \&\ \psi) \rightarrow \chi)\ \mathit{in}\ v]$
  by *PLM-solver*

lemma *oth-class-taut-9-a*[*PLM*]:
  $[(\varphi\ \&\ \psi) \rightarrow \varphi\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-9-b*[*PLM*]:
  $[(\varphi\ \&\ \psi) \rightarrow \psi\ \mathit{in}\ v]$
  by *PLM-solver*

lemma *oth-class-taut-10-a*[*PLM*]:
  $[\varphi \rightarrow (\psi \rightarrow (\varphi\ \&\ \psi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-10-b*[*PLM*]:
  $[(\varphi \rightarrow (\psi \rightarrow \chi)) \equiv (\psi \rightarrow (\varphi \rightarrow \chi))\ \mathit{in}\ v]$
  by *PLM-solver*
lemma *oth-class-taut-10-c*[*PLM*]:

$[(\varphi \to \psi) \to ((\varphi \to \chi) \to (\varphi \to (\psi \;\&\; \chi)))$ *in v*]
by *PLM-solver*
lemma *oth-class-taut-10-d*[*PLM*]:
$[(\varphi \to \chi) \to ((\psi \to \chi) \to ((\varphi \lor \psi) \to \chi))$ *in v*]
by *PLM-solver*
lemma *oth-class-taut-10-e*[*PLM*]:
$[(\varphi \to \psi) \to ((\chi \to \Theta) \to ((\varphi \;\&\; \chi) \to (\psi \;\&\; \Theta)))$ *in v*]
by *PLM-solver*
lemma *oth-class-taut-10-f*[*PLM*]:
$[((\varphi \;\&\; \psi) \equiv (\varphi \;\&\; \chi)) \equiv (\varphi \to (\psi \equiv \chi))$ *in v*]
by *PLM-solver*
lemma *oth-class-taut-10-g*[*PLM*]:
$[((\varphi \;\&\; \psi) \equiv (\chi \;\&\; \psi)) \equiv (\psi \to (\varphi \equiv \chi))$ *in v*]
by *PLM-solver*

attribute-setup *equiv-lr* = ⟪
Scan.succeed (*Thm.rule-attribute* []
(*fn - => fn thm => thm RS* @{*thm* ≡*E*(*1*)}))
⟫

attribute-setup *equiv-rl* = ⟪
Scan.succeed (*Thm.rule-attribute* []
(*fn - => fn thm => thm RS* @{*thm* ≡*E*(*2*)}))
⟫

attribute-setup *equiv-sym* = ⟪
Scan.succeed (*Thm.rule-attribute* []
(*fn - => fn thm => thm RS* @{*thm oth-class-taut-3-g*[*equiv-lr*]}))
⟫

attribute-setup *conj1* = ⟪
Scan.succeed (*Thm.rule-attribute* []
(*fn - => fn thm => thm RS* @{*thm* &*E*(*1*)}))
⟫

attribute-setup *conj2* = ⟪
Scan.succeed (*Thm.rule-attribute* []
(*fn - => fn thm => thm RS* @{*thm* &*E*(*2*)}))
⟫

attribute-setup *conj-sym* = ⟪
Scan.succeed (*Thm.rule-attribute* []
(*fn - => fn thm => thm RS* @{*thm oth-class-taut-3-b*[*equiv-lr*]}))
⟫

9.7 Identity

lemma *id-eq-prop-prop-1*[*PLM*]:
$[(F{::}\Pi_1) = F$ *in v*]
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-2*[*PLM*]:
$[((F{::}\Pi_1) = G) \to (G = F)$ *in v*]
by (*meson id-eq-prop-prop-1 CP ded-thm-cor-3 l-identity*[*axiom-instance*])
lemma *id-eq-prop-prop-3*[*PLM*]:
$[(((F{::}\Pi_1) = G) \;\&\; (G = H)) \to (F = H)$ *in v*]
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* &*E*)
lemma *id-eq-prop-prop-4-a*[*PLM*]:
$[(F{::}\Pi_2) = F$ *in v*]
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-4-b*[*PLM*]:
$[(F{::}\Pi_3) = F$ *in v*]
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-5-a*[*PLM*]:

$[((F::\Pi_2) = G) \rightarrow (G = F)$ *in v*$]$
by (*meson id-eq-prop-prop-4-a CP ded-thm-cor-3 l-identity*[*axiom-instance*])
lemma *id-eq-prop-prop-5-b*[*PLM*]:
$[((F::\Pi_3) = G) \rightarrow (G = F)$ *in v*$]$
by (*meson id-eq-prop-prop-4-b CP ded-thm-cor-3 l-identity*[*axiom-instance*])
lemma *id-eq-prop-prop-6-a*[*PLM*]:
$[(((F::\Pi_2) = G)$ **&** $(G = H)) \rightarrow (F = H)$ *in v*$]$
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)
lemma *id-eq-prop-prop-6-b*[*PLM*]:
$[(((F::\Pi_3) = G)$ **&** $(G = H)) \rightarrow (F = H)$ *in v*$]$
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)
lemma *id-eq-prop-prop-7*[*PLM*]:
$[(p::\Pi_0) = p$ *in v*$]$
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-7-b*[*PLM*]:
$[(p::o) = p$ *in v*$]$
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-8*[*PLM*]:
$[((p::\Pi_0) = q) \rightarrow (q = p)$ *in v*$]$
by (*meson id-eq-prop-prop-7 CP ded-thm-cor-3 l-identity*[*axiom-instance*])
lemma *id-eq-prop-prop-8-b*[*PLM*]:
$[((p::o) = q) \rightarrow (q = p)$ *in v*$]$
by (*meson id-eq-prop-prop-7-b CP ded-thm-cor-3 l-identity*[*axiom-instance*])
lemma *id-eq-prop-prop-9*[*PLM*]:
$[(((p::\Pi_0) = q)$ **&** $(q = r)) \rightarrow (p = r)$ *in v*$]$
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)
lemma *id-eq-prop-prop-9-b*[*PLM*]:
$[(((p::o) = q)$ **&** $(q = r)) \rightarrow (p = r)$ *in v*$]$
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)

lemma *eq-E-simple-1*[*PLM*]:
$[(x =_E y) \equiv ((\!|O!,x|\!)$ **&** $(\!|O!,y|\!)$ **&** $\Box(\forall F . (\!|F,x|\!) \equiv (\!|F,y|\!)))$ *in v*$]$
proof (*rule* $\equiv$*I*; *rule CP*)
assume *1*: $[x =_E y$ *in v*$]$
have $[\forall\ x\ y . ((x^P) =_E (y^P)) \equiv ((\!|O!,x^P|\!)$ **&** $(\!|O!,y^P|\!)$
& $\Box(\forall F . (\!|F,x^P|\!) \equiv (\!|F,y^P|\!)))$ *in v*$]$
unfolding *identity$_E$-infix-def identity$_E$-def*
apply (*rule lambda-predicates-2-2*[*axiom-universal*, *axiom-universal*, *axiom-instance*])
by *show-proper*
moreover have $[\exists\ \alpha . (\alpha^P) = x$ *in v*$]$
apply (*rule cqt-5-mod*[**where** $\psi{=}\lambda\ x . x =_E y$, *axiom-instance*,*deduction*])
unfolding *identity$_E$-infix-def*
apply (*rule SimpleExOrEnc.intros*)
using *1* **unfolding** *identity$_E$-infix-def* **by** *auto*
moreover have $[\exists\ \beta . (\beta^P) = y$ *in v*$]$
apply (*rule cqt-5-mod*[**where** $\psi{=}\lambda\ y . x =_E y$,*axiom-instance*,*deduction*])
unfolding *identity$_E$-infix-def*
apply (*rule SimpleExOrEnc.intros*) **using** *1*
unfolding *identity$_E$-infix-def* **by** *auto*
ultimately have $[(x =_E y) \equiv ((\!|O!,x|\!)$ **&** $(\!|O!,y|\!)$
& $\Box(\forall F . (\!|F,x|\!) \equiv (\!|F,y|\!)))$ *in v*$]$
using *cqt-1-κ*[*axiom-instance*,*deduction*, *deduction*] **by** *meson*
thus $[((\!|O!,x|\!)$ **&** $(\!|O!,y|\!)$ **&** $\Box(\forall F . (\!|F,x|\!) \equiv (\!|F,y|\!)))$ *in v*$]$
using *1* $\equiv$*E*(*1*) **by** *blast*
next
assume *1*: $[(\!|O!,x|\!)$ **&** $(\!|O!,y|\!)$ **&** $\Box(\forall F. (\!|F,x|\!) \equiv (\!|F,y|\!))$ *in v*$]$
have $[\forall\ x\ y . ((x^P) =_E (y^P)) \equiv ((\!|O!,x^P|\!)$ **&** $(\!|O!,y^P|\!)$
& $\Box(\forall F . (\!|F,x^P|\!) \equiv (\!|F,y^P|\!)))$ *in v*$]$
unfolding *identity$_E$-def identity$_E$-infix-def*
apply (*rule lambda-predicates-2-2*[*axiom-universal*, *axiom-universal*, *axiom-instance*])
by *show-proper*
moreover have $[\exists\ \alpha . (\alpha^P) = x$ *in v*$]$
apply (*rule cqt-5-mod*[**where** $\psi{=}\lambda\ x . (\!|O!,x|\!)$,*axiom-instance*,*deduction*])

apply (*rule SimpleExOrEnc.intros*)
using *1*[*conj1*,*conj1*] **by** *auto*
moreover have [$\exists\ \beta\ .\ (\beta^P) = y$ *in v*]
apply (*rule cqt-5-mod*[**where** ψ=$\lambda\ y\ .\ (|O!,y|)$,*axiom-instance*,*deduction*])
apply (*rule SimpleExOrEnc.intros*)
using *1*[*conj1*,*conj2*] **by** *auto*
ultimately have [$(x =_E y) \equiv ((|O!,x|)$ **&** $(|O!,y|)$
& $\Box(\forall\ F\ .\ (|F,x|) \equiv (|F,y|)))$ *in v*]
using *cqt-1-κ*[*axiom-instance*,*deduction*, *deduction*] **by** *meson*
thus [$(x =_E y)$ *in v*] **using** *1* $\equiv E(2)$ **by** *blast*
qed
lemma *eq-E-simple-2*[*PLM*]:
[$(x =_E y) \rightarrow (x = y)$ *in v*]
unfolding *identity-defs* **by** *PLM-solver*
lemma *eq-E-simple-3*[*PLM*]:
[$(x = y) \equiv (((|O!,x|)$ **&** $(|O!,y|)$ **&** $\Box(\forall\ F\ .\ (|F,x|) \equiv (|F,y|)))$
$\vee ((|A!,x|)$ **&** $(|A!,y|)$ **&** $\Box(\forall\ F.\ \{|x,F|\} \equiv \{|y,F|\})))$ *in v*]
using *eq-E-simple-1*
apply − **unfolding** *identity-defs*
by *PLM-solver*

lemma *id-eq-obj-1*[*PLM*]: [$(x^P) = (x^P)$ *in v*]
proof −
have [$(\Diamond(|E!,\ x^P|)) \vee (\neg\Diamond(|E!,\ x^P|))$ *in v*]
using *PLM.oth-class-taut-2* **by** *simp*
hence [$(\Diamond(|E!,\ x^P|))$ *in v*] $\vee$ [$(\neg\Diamond(|E!,\ x^P|))$ *in v*]
using *CP* $\vee E(1)$ **by** *blast*
moreover {
assume [$(\Diamond(|E!,\ x^P|))$ *in v*]
hence [$(|\boldsymbol{\lambda}x.\ \Diamond(|E!,x^P|),x^P|)$ *in v*]
apply (*rule lambda-predicates-2-1*[*axiom-instance*, *equiv-rl*, *rotated*])
by *show-proper*
hence [$(|\boldsymbol{\lambda}x.\ \Diamond(|E!,x^P|),x^P|)$ **&** $(|\boldsymbol{\lambda}x.\ \Diamond(|E!,x^P|),x^P|)$
& $\Box(\forall\ F.\ (|F,x^P|) \equiv (|F,x^P|))$ *in v*]
apply − **by** *PLM-solver*
hence [$(x^P) =_E (x^P)$ *in v*]
using *eq-E-simple-1*[*equiv-rl*] **unfolding** *Ordinary-def* **by** *fast*
}
moreover {
assume [$(\neg\Diamond(|E!,\ x^P|))$ *in v*]
hence [$(|\boldsymbol{\lambda}x.\ \neg\Diamond(|E!,x^P|),x^P|)$ *in v*]
apply (*rule lambda-predicates-2-1*[*axiom-instance*, *equiv-rl*, *rotated*])
by *show-proper*
hence [$(|\boldsymbol{\lambda}x.\ \neg\Diamond(|E!,x^P|),x^P|)$ **&** $(|\boldsymbol{\lambda}x.\ \neg\Diamond(|E!,x^P|),x^P|)$
& $\Box(\forall\ F.\ \{|x^P,F|\} \equiv \{|x^P,F|\})$ *in v*]
apply − **by** *PLM-solver*
}
ultimately show *?thesis* **unfolding** *identity-defs Ordinary-def Abstract-def*
using $\vee I$ **by** *blast*
qed
lemma *id-eq-obj-2*[*PLM*]:
[$((x^P) = (y^P)) \rightarrow ((y^P) = (x^P))$ *in v*]
by (*meson l-identity*[*axiom-instance*] *id-eq-obj-1 CP ded-thm-cor-3*)
lemma *id-eq-obj-3*[*PLM*]:
[$((x^P) = (y^P))$ **&** $((y^P) = (z^P)) \rightarrow ((x^P) = (z^P))$ *in v*]
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)
end

Remark 21. *To unify the statements of the properties of equality a type class is introduced.*

class *id-eq* = *quantifiable-and-identifiable* +
assumes *id-eq-1*: [$(x :: 'a) = x$ *in v*]
assumes *id-eq-2*: [$((x :: 'a) = y) \rightarrow (y = x)$ *in v*]
assumes *id-eq-3*: [$((x :: 'a) = y)$ **&** $(y = z) \rightarrow (x = z)$ *in v*]

```
instantiation ν :: id-eq
begin
  instance proof
    fix x :: ν and v
    show [x = x in v]
      using PLM.id-eq-obj-1
      by (simp add: identity-ν-def)
  next
    fix x y::ν and v
    show [x = y → y = x in v]
      using PLM.id-eq-obj-2
      by (simp add: identity-ν-def)
  next
    fix x y z::ν and v
    show [((x = y) & (y = z)) → x = z in v]
      using PLM.id-eq-obj-3
      by (simp add: identity-ν-def)
  qed
end

instantiation o :: id-eq
begin
  instance proof
    fix x :: o and v
    show [x = x in v]
      using PLM.id-eq-prop-prop-7 .
  next
    fix x y :: o and v
    show [x = y → y = x in v]
      using PLM.id-eq-prop-prop-8 .
  next
    fix x y z :: o and v
    show [((x = y) & (y = z)) → x = z in v]
      using PLM.id-eq-prop-prop-9 .
  qed
end

instantiation Π₁ :: id-eq
begin
  instance proof
    fix x :: Π₁ and v
    show [x = x in v]
      using PLM.id-eq-prop-prop-1 .
  next
    fix x y :: Π₁ and v
    show [x = y → y = x in v]
      using PLM.id-eq-prop-prop-2 .
  next
    fix x y z :: Π₁ and v
    show [((x = y) & (y = z)) → x = z in v]
      using PLM.id-eq-prop-prop-3 .
  qed
end

instantiation Π₂ :: id-eq
begin
  instance proof
    fix x :: Π₂ and v
    show [x = x in v]
      using PLM.id-eq-prop-prop-4-a .
  next
    fix x y :: Π₂ and v
```

```
    show [x = y → y = x in v]
      using PLM.id-eq-prop-prop-5-a .
  next
    fix x y z :: Π₂ and v
    show [((x = y) & (y = z)) → x = z in v]
      using PLM.id-eq-prop-prop-6-a .
  qed
end

instantiation Π₃ :: id-eq
begin
  instance proof
    fix x :: Π₃ and v
    show [x = x in v]
      using PLM.id-eq-prop-prop-4-b .
  next
    fix x y :: Π₃ and v
    show [x = y → y = x in v]
      using PLM.id-eq-prop-prop-5-b .
  next
    fix x y z :: Π₃ and v
    show [((x = y) & (y = z)) → x = z in v]
      using PLM.id-eq-prop-prop-6-b .
  qed
end

context PLM
begin
  lemma id-eq-1[PLM]:
    [(x::'a::id-eq) = x in v]
    using id-eq-1 .
  lemma id-eq-2[PLM]:
    [((x::'a::id-eq) = y) → (y = x) in v]
    using id-eq-2 .
  lemma id-eq-3[PLM]:
    [((x::'a::id-eq) = y) & (y = z) → (x = z) in v]
    using id-eq-3 .

  attribute-setup eq-sym = ⟪
    Scan.succeed (Thm.rule-attribute []
      (fn - => fn thm => thm RS @{thm id-eq-2[deduction]}))
  ⟫


  lemma all-self-eq-1[PLM]:
    [□(∀ α :: 'a::id-eq . α = α) in v]
    by PLM-solver
  lemma all-self-eq-2[PLM]:
    [∀ α :: 'a::id-eq . □(α = α) in v]
    by PLM-solver

  lemma t-id-t-proper-1[PLM]:
    [τ = τ' → (∃ β . (β^P) = τ) in v]
    proof (rule CP)
      assume [τ = τ' in v]
      moreover {
        assume [τ =_E τ' in v]
        hence [∃ β . (β^P) = τ in v]
          apply −
          apply (rule cqt-5-mod[where ψ=λ τ . τ =_E τ', axiom-instance, deduction])
           subgoal unfolding identity-defs by (rule SimpleExOrEnc.intros)
          by simp
      }
```

moreover {
assume $[(\!|A!,\tau|\!) \ \&\ (\!|A!,\tau'|\!) \ \&\ \Box(\forall F.\ \{\!|\tau,F|\!\} \equiv \{\!|\tau',F|\!\})\ in\ v]$
hence $[\exists\ \beta\ .\ (\beta^P) = \tau\ in\ v]$
apply $-$
apply (*rule cqt-5-mod*[**where** $\psi = \lambda\ \tau\ .\ (\!|A!,\tau|\!)$, *axiom-instance*, *deduction*])
subgoal unfolding *identity-defs* **by** (*rule SimpleExOrEnc.intros*)
by *PLM-solver*
}
ultimately show $[\exists\ \beta\ .\ (\beta^P) = \tau\ in\ v]$ **unfolding** *identity*$_\kappa$*-def*
using *intro-elim-4-b reductio-aa-1* **by** *blast*
qed

lemma *t-id-t-proper-2*[*PLM*]: $[\tau = \tau' \rightarrow (\exists\ \beta\ .\ (\beta^P) = \tau')\ in\ v]$
proof (*rule CP*)
assume $[\tau = \tau'\ in\ v]$
moreover {
assume $[\tau =_E \tau'\ in\ v]$
hence $[\exists\ \beta\ .\ (\beta^P) = \tau'\ in\ v]$
apply $-$
apply (*rule cqt-5-mod*[**where** $\psi = \lambda\ \tau'\ .\ \tau =_E \tau'$, *axiom-instance*, *deduction*])
subgoal unfolding *identity-defs* **by** (*rule SimpleExOrEnc.intros*)
by *simp*
}
moreover {
assume $[(\!|A!,\tau|\!) \ \&\ (\!|A!,\tau'|\!) \ \&\ \Box(\forall F.\ \{\!|\tau,F|\!\} \equiv \{\!|\tau',F|\!\})\ in\ v]$
hence $[\exists\ \beta\ .\ (\beta^P) = \tau'\ in\ v]$
apply $-$
apply (*rule cqt-5-mod*[**where** $\psi = \lambda\ \tau\ .\ (\!|A!,\tau|\!)$, *axiom-instance*, *deduction*])
subgoal unfolding *identity-defs* **by** (*rule SimpleExOrEnc.intros*)
by *PLM-solver*
}
ultimately show $[\exists\ \beta\ .\ (\beta^P) = \tau'\ in\ v]$ **unfolding** *identity*$_\kappa$*-def*
using *intro-elim-4-b reductio-aa-1* **by** *blast*
qed

lemma *id-nec*[*PLM*]: $[((\alpha{::}'a{::}id\text{-}eq) = (\beta)) \equiv \Box((\alpha) = (\beta))\ in\ v]$
apply (*rule* $\equiv I$)
using *l-identity*[**where** $\varphi = (\lambda\ \beta\ .\ \Box((\alpha) = (\beta)))$, *axiom-instance*]
id-eq-1 RN ded-thm-cor-4 **unfolding** *identity-ν-def*
apply *blast*
using *qml-2*[*axiom-instance*] **by** *blast*

lemma *id-nec-desc*[*PLM*]:
$[((\iota x.\ \varphi\ x) = (\iota x.\ \psi\ x)) \equiv \Box((\iota x.\ \varphi\ x) = (\iota x.\ \psi\ x))\ in\ v]$
proof (*cases* $[(\exists\ \alpha.\ (\alpha^P) = (\iota x\ .\ \varphi\ x))\ in\ v] \wedge [(\exists\ \beta.\ (\beta^P) = (\iota x\ .\ \psi\ x))\ in\ v]$)
assume $[(\exists\ \alpha.\ (\alpha^P) = (\iota x\ .\ \varphi\ x))\ in\ v] \wedge [(\exists\ \beta.\ (\beta^P) = (\iota x\ .\ \psi\ x))\ in\ v]$
then obtain α **and** β **where**
$[(\alpha^P) = (\iota x\ .\ \varphi\ x)\ in\ v] \wedge [(\beta^P) = (\iota x\ .\ \psi\ x)\ in\ v]$
apply $-$ **unfolding** *conn-defs* **by** *PLM-solver*
moreover {
moreover have $[(\alpha) = (\beta) \equiv \Box((\alpha) = (\beta))\ in\ v]$ **by** *PLM-solver*
ultimately have $[((\iota x.\ \varphi\ x) = (\beta^P) \equiv \Box((\iota x.\ \varphi\ x) = (\beta^P)))\ in\ v]$
using *l-identity*[**where** $\varphi = \lambda\ \alpha\ .\ (\alpha) = (\beta^P) \equiv \Box((\alpha) = (\beta^P))$, *axiom-instance*]
modus-ponens **unfolding** *identity-ν-def* **by** *metis*
}
ultimately show *?thesis*
using *l-identity*[**where** $\varphi = \lambda\ \alpha\ .\ (\iota x\ .\ \varphi\ x) = (\alpha)$
$\equiv \Box((\iota x\ .\ \varphi\ x) = (\alpha))$, *axiom-instance*]
modus-ponens **by** *metis*
next
assume $\neg([(\exists\ \alpha.\ (\alpha^P) = (\iota x\ .\ \varphi\ x))\ in\ v] \wedge [(\exists\ \beta.\ (\beta^P) = (\iota x\ .\ \psi\ x))\ in\ v])$
hence $\neg[(\!|A!,(\iota x\ .\ \varphi\ x)|\!)\ in\ v] \wedge \neg[(\iota x\ .\ \varphi\ x) =_E (\iota x\ .\ \psi\ x)\ in\ v]$
$\vee \neg[(\!|A!,(\iota x\ .\ \psi\ x)|\!)\ in\ v] \wedge \neg[(\iota x\ .\ \varphi\ x) =_E (\iota x\ .\ \psi\ x)\ in\ v]$

unfolding *identity*$_E$*-infix-def*
using *cqt-5*[*axiom-instance*] *PLM.contraposition-1 SimpleExOrEnc.intros vdash-properties-10* **by** *meson*
hence $\neg[(\iota x \ . \ \varphi \ x) = (\iota x \ . \ \psi \ x) \ in \ v]$
apply − **unfolding** *identity-defs* **by** *PLM-solver*
thus *?thesis* **apply** − **apply** *PLM-solver*
using *qml-2*[*axiom-instance*, *deduction*] **by** *auto*
qed

9.8 Quantification

lemma *rule-ui*[*PLM*,*PLM-elim*,*PLM-dest*]:
$[\forall \alpha \ . \ \varphi \ \alpha \ in \ v] \Longrightarrow [\varphi \ \beta \ in \ v]$
by (*meson cqt-1*[*axiom-instance*, *deduction*])
lemmas $\forall E$ = *rule-ui*

lemma *rule-ui-2*[*PLM*,*PLM-elim*,*PLM-dest*]:
$[\![[\forall \alpha \ . \ \varphi \ (\alpha^P) \ in \ v]; [\exists \ \alpha \ . \ (\alpha)^P = \beta \ in \ v]]\!] \Longrightarrow [\varphi \ \beta \ in \ v]$
using *cqt-1-*κ[*axiom-instance*, *deduction*, *deduction*] **by** *blast*

lemma *cqt-orig-1*[*PLM*]:
$[(\forall \alpha. \ \varphi \ \alpha) \rightarrow \varphi \ \beta \ in \ v]$
by *PLM-solver*
lemma *cqt-orig-2*[*PLM*]:
$[(\forall \alpha. \ \varphi \rightarrow \psi \ \alpha) \rightarrow (\varphi \rightarrow (\forall \alpha. \ \psi \ \alpha)) \ in \ v]$
by *PLM-solver*

lemma *universal*[*PLM*]:
$(\bigwedge \alpha \ . \ [\varphi \ \alpha \ in \ v]) \Longrightarrow [\forall \ \alpha \ . \ \varphi \ \alpha \ in \ v]$
using *rule-gen* **.**
lemmas $\forall I$ = *universal*

lemma *cqt-basic-1*[*PLM*]:
$[(\forall \alpha. \ (\forall \beta \ . \ \varphi \ \alpha \ \beta)) \equiv (\forall \beta. \ (\forall \alpha. \ \varphi \ \alpha \ \beta)) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-2*[*PLM*]:
$[(\forall \alpha. \ \varphi \ \alpha \equiv \psi \ \alpha) \equiv ((\forall \alpha. \ \varphi \ \alpha \rightarrow \psi \ \alpha) \ \& \ (\forall \alpha. \ \psi \ \alpha \rightarrow \varphi \ \alpha)) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-3*[*PLM*]:
$[(\forall \alpha. \ \varphi \ \alpha \equiv \psi \ \alpha) \rightarrow ((\forall \alpha. \ \varphi \ \alpha) \equiv (\forall \alpha. \ \psi \ \alpha)) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-4*[*PLM*]:
$[(\forall \alpha. \ \varphi \ \alpha \ \& \ \psi \ \alpha) \equiv ((\forall \alpha. \ \varphi \ \alpha) \ \& \ (\forall \alpha. \ \psi \ \alpha)) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-6*[*PLM*]:
$[(\forall \alpha. \ (\forall \alpha. \ \varphi \ \alpha)) \equiv (\forall \alpha. \ \varphi \ \alpha) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-7*[*PLM*]:
$[(\varphi \rightarrow (\forall \alpha \ . \ \psi \ \alpha)) \equiv (\forall \alpha.(\varphi \rightarrow \psi \ \alpha)) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-8*[*PLM*]:
$[((\forall \alpha. \ \varphi \ \alpha) \lor (\forall \alpha. \ \psi \ \alpha)) \rightarrow (\forall \alpha. \ (\varphi \ \alpha \lor \psi \ \alpha)) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-9*[*PLM*]:
$[((\forall \alpha. \ \varphi \ \alpha \rightarrow \psi \ \alpha) \ \& \ (\forall \alpha. \ \psi \ \alpha \rightarrow \chi \ \alpha)) \rightarrow (\forall \alpha. \ \varphi \ \alpha \rightarrow \chi \ \alpha) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-10*[*PLM*]:
$[((\forall \alpha. \ \varphi \ \alpha \equiv \psi \ \alpha) \ \& \ (\forall \alpha. \ \psi \ \alpha \equiv \chi \ \alpha)) \rightarrow (\forall \alpha. \ \varphi \ \alpha \equiv \chi \ \alpha) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-11*[*PLM*]:
$[(\forall \alpha. \ \varphi \ \alpha \equiv \psi \ \alpha) \equiv (\forall \alpha. \ \psi \ \alpha \equiv \varphi \ \alpha) \ in \ v]$
by *PLM-solver*
lemma *cqt-basic-12*[*PLM*]:

```
  [(∀ α. φ α) ≡ (∀ β. φ β) in v]
  by PLM-solver

lemma existential[PLM,PLM-intro]:
  [φ α in v] ⟹ [∃ α. φ α in v]
  unfolding exists-def by PLM-solver
lemmas ∃ I = existential
lemma instantiation-[PLM,PLM-elim,PLM-dest]:
  ⟦[∃ α . φ α in v]; (⋀α.[φ α in v] ⟹ [ψ in v])⟧ ⟹ [ψ in v]
  unfolding exists-def by PLM-solver

lemma Instantiate:
  assumes [∃ x . φ x  in v]
  obtains x where [φ x in v]
  apply (insert assms) unfolding exists-def by PLM-solver
lemmas ∃ E = Instantiate

lemma cqt-further-1[PLM]:
  [(∀ α. φ α) → (∃ α. φ α) in v]
  by PLM-solver
lemma cqt-further-2[PLM]:
  [(¬(∀ α. φ α)) ≡ (∃ α. ¬φ α) in v]
  unfolding exists-def by PLM-solver
lemma cqt-further-3[PLM]:
  [(∀ α. φ α) ≡ ¬(∃ α. ¬φ α) in v]
  unfolding exists-def by PLM-solver
lemma cqt-further-4[PLM]:
  [(¬(∃ α. φ α)) ≡ (∀ α. ¬φ α) in v]
  unfolding exists-def by PLM-solver
lemma cqt-further-5[PLM]:
  [(∃ α. φ α & ψ α) → ((∃ α. φ α) & (∃ α. ψ α)) in v]
    unfolding exists-def by PLM-solver
lemma cqt-further-6[PLM]:
  [(∃ α. φ α ∨ ψ α) ≡ ((∃ α. φ α) ∨ (∃ α. ψ α)) in v]
  unfolding exists-def by PLM-solver
lemma cqt-further-10[PLM]:
  [(φ (α::'a::id-eq) & (∀ β . φ β → β = α)) ≡ (∀ β . φ β ≡ β = α) in v]
  apply PLM-solver
   using l-identity[axiom-instance, deduction, deduction] id-eq-2[deduction]
   apply blast
  using id-eq-1 by auto
lemma cqt-further-11[PLM]:
  [((∀ α. φ α) & (∀ α. ψ α)) → (∀ α. φ α ≡ ψ α) in v]
  by PLM-solver
lemma cqt-further-12[PLM]:
  [((¬(∃ α. φ α)) & (¬(∃ α. ψ α))) → (∀ α. φ α ≡ ψ α) in v]
  unfolding exists-def by PLM-solver
lemma cqt-further-13[PLM]:
  [((∃ α. φ α) & (¬(∃ α. ψ α))) → (¬(∀ α. φ α ≡ ψ α)) in v]
  unfolding exists-def by PLM-solver
lemma cqt-further-14[PLM]:
  [(∃ α. ∃ β. φ α β) ≡ (∃ β. ∃ α. φ α β) in v]
  unfolding exists-def by PLM-solver

lemma nec-exist-unique[PLM]:
  [(∀ x. φ x → □(φ x)) → ((∃ !x. φ x) → (∃ !x. □(φ x))) in v]
  proof (rule CP)
    assume a: [∀ x. φ x → □φ x in v]
    show [(∃ !x. φ x) → (∃ !x. □φ x) in v]
    proof (rule CP)
      assume [(∃ !x. φ x) in v]
      hence [∃ α. φ α & (∀ β. φ β → β = α) in v]
        by (simp only: exists-unique-def)
```

```
      then obtain α where 1:
        [φ α & (∀ β. φ β → β = α) in v]
        by (rule ∃ E)
      {
        fix β
        have [□φ β → β = α in v]
          using 1 &E(2) qml-2[axiom-instance]
            ded-thm-cor-3 ∀ E by fastforce
      }
      hence [∀ β. □φ β → β = α in v] by (rule ∀ I)
      moreover have [□(φ α) in v]
        using 1 &E(1) a vdash-properties-10 cqt-orig-1[deduction]
        by fast
      ultimately have [∃ α. □(φ α) & (∀ β. □φ β → β = α) in v]
        using &I ∃ I by fast
      thus [(∃ !x. □φ x) in v]
        unfolding exists-unique-def by assumption
    qed
  qed
```

9.9 Actuality and Descriptions

```
lemma nec-imp-act[PLM]: [□φ → 𝓐φ in v]
  apply (rule CP)
  using qml-act-2[axiom-instance, equiv-lr]
        qml-2[axiom-actualization, axiom-instance]
        logic-actual-nec-2[axiom-instance, equiv-lr, deduction]
  by blast
lemma act-conj-act-1[PLM]:
  [𝓐(𝓐φ → φ) in v]
  using equiv-def logic-actual-nec-2[axiom-instance]
        logic-actual-nec-4[axiom-instance] &E(2) ≡E(2)
  by metis
lemma act-conj-act-2[PLM]:
  [𝓐(φ → 𝓐φ) in v]
  using logic-actual-nec-2[axiom-instance] qml-act-1[axiom-instance]
        ded-thm-cor-3 ≡E(2) nec-imp-act
  by blast
lemma act-conj-act-3[PLM]:
  [(𝓐φ & 𝓐ψ) → 𝓐(φ & ψ) in v]
  unfolding conn-defs
  by (metis logic-actual-nec-2[axiom-instance]
            logic-actual-nec-1[axiom-instance]
            ≡E(2) CP ≡E(4) reductio-aa-2
            vdash-properties-10)
lemma act-conj-act-4[PLM]:
  [𝓐(𝓐φ ≡ φ) in v]
  unfolding equiv-def
  by (PLM-solver PLM-intro: act-conj-act-3[where φ=𝓐φ → φ
                          and ψ=φ → 𝓐φ, deduction])
lemma closure-act-1a[PLM]:
  [𝓐𝓐(𝓐φ ≡ φ) in v]
  using logic-actual-nec-4[axiom-instance]
        act-conj-act-4 ≡E(1)
  by blast
lemma closure-act-1b[PLM]:
  [𝓐𝓐𝓐(𝓐φ ≡ φ) in v]
  using logic-actual-nec-4[axiom-instance]
        act-conj-act-4 ≡E(1)
  by blast
lemma closure-act-1c[PLM]:
  [𝓐𝓐𝓐𝓐(𝓐φ ≡ φ) in v]
  using logic-actual-nec-4[axiom-instance]
```

act-conj-act-4 $\equiv E(1)$
by *blast*
lemma *closure-act-2*[*PLM*]:
$[\forall\,\alpha.\ \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{A}}(\varphi\ \alpha) \equiv \varphi\ \alpha)\ in\ v]$
by *PLM-solver*

lemma *closure-act-3*[*PLM*]:
$[\boldsymbol{\mathcal{A}}(\forall\,\alpha.\ \boldsymbol{\mathcal{A}}(\varphi\ \alpha) \equiv \varphi\ \alpha)\ in\ v]$
by (*PLM-solver PLM-intro*: *logic-actual-nec-3*[*axiom-instance*, *equiv-rl*])
lemma *closure-act-4*[*PLM*]:
$[\boldsymbol{\mathcal{A}}(\forall\,\alpha_1\ \alpha_2.\ \boldsymbol{\mathcal{A}}(\varphi\ \alpha_1\ \alpha_2) \equiv \varphi\ \alpha_1\ \alpha_2)\ in\ v]$
by (*PLM-solver PLM-intro*: *logic-actual-nec-3*[*axiom-instance*, *equiv-rl*])
lemma *closure-act-4-b*[*PLM*]:
$[\boldsymbol{\mathcal{A}}(\forall\,\alpha_1\ \alpha_2\ \alpha_3.\ \boldsymbol{\mathcal{A}}(\varphi\ \alpha_1\ \alpha_2\ \alpha_3) \equiv \varphi\ \alpha_1\ \alpha_2\ \alpha_3)\ in\ v]$
by (*PLM-solver PLM-intro*: *logic-actual-nec-3*[*axiom-instance*, *equiv-rl*])
lemma *closure-act-4-c*[*PLM*]:
$[\boldsymbol{\mathcal{A}}(\forall\,\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4.\ \boldsymbol{\mathcal{A}}(\varphi\ \alpha_1\ \alpha_2\ \alpha_3\ \alpha_4) \equiv \varphi\ \alpha_1\ \alpha_2\ \alpha_3\ \alpha_4)\ in\ v]$
by (*PLM-solver PLM-intro*: *logic-actual-nec-3*[*axiom-instance*, *equiv-rl*])

lemma *RA*[*PLM*,*PLM-intro*]:
$([\varphi\ in\ dw]) \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi\ in\ dw]$
using *logic-actual*[*necessitation-averse-axiom-instance*, *equiv-rl*] **.**

lemma *RA-2*[*PLM*,*PLM-intro*]:
$([\psi\ in\ dw] \Longrightarrow [\varphi\ in\ dw]) \Longrightarrow ([\boldsymbol{\mathcal{A}}\psi\ in\ dw] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi\ in\ dw])$
using *RA logic-actual*[*necessitation-averse-axiom-instance*] *intro-elim-6-a* **by** *blast*

context
begin
private lemma *ActualE*[*PLM*,*PLM-elim*,*PLM-dest*]:
$[\boldsymbol{\mathcal{A}}\varphi\ in\ dw] \Longrightarrow [\varphi\ in\ dw]$
using *logic-actual*[*necessitation-averse-axiom-instance*, *equiv-lr*] **.**

private lemma *NotActualD*[*PLM-dest*]:
$\neg[\boldsymbol{\mathcal{A}}\varphi\ in\ dw] \Longrightarrow \neg[\varphi\ in\ dw]$
using *RA* **by** *metis*

private lemma *ActualImplI*[*PLM-intro*]:
$[\boldsymbol{\mathcal{A}}\varphi \rightarrow \boldsymbol{\mathcal{A}}\psi\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}(\varphi \rightarrow \psi)\ in\ v]$
using *logic-actual-nec-2*[*axiom-instance*, *equiv-rl*] **.**
private lemma *ActualImplE*[*PLM-dest*, *PLM-elim*]:
$[\boldsymbol{\mathcal{A}}(\varphi \rightarrow \psi)\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi \rightarrow \boldsymbol{\mathcal{A}}\psi\ in\ v]$
using *logic-actual-nec-2*[*axiom-instance*, *equiv-lr*] **.**
private lemma *NotActualImplD*[*PLM-dest*]:
$\neg[\boldsymbol{\mathcal{A}}(\varphi \rightarrow \psi)\ in\ v] \Longrightarrow \neg[\boldsymbol{\mathcal{A}}\varphi \rightarrow \boldsymbol{\mathcal{A}}\psi\ in\ v]$
using *ActualImplI* **by** *blast*

private lemma *ActualNotI*[*PLM-intro*]:
$[\neg\boldsymbol{\mathcal{A}}\varphi\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\neg\varphi\ in\ v]$
using *logic-actual-nec-1*[*axiom-instance*, *equiv-rl*] **.**
lemma *ActualNotE*[*PLM-elim*,*PLM-dest*]:
$[\boldsymbol{\mathcal{A}}\neg\varphi\ in\ v] \Longrightarrow [\neg\boldsymbol{\mathcal{A}}\varphi\ in\ v]$
using *logic-actual-nec-1*[*axiom-instance*, *equiv-lr*] **.**
lemma *NotActualNotD*[*PLM-dest*]:
$\neg[\boldsymbol{\mathcal{A}}\neg\varphi\ in\ v] \Longrightarrow \neg[\neg\boldsymbol{\mathcal{A}}\varphi\ in\ v]$
using *ActualNotI* **by** *blast*

private lemma *ActualConjI*[*PLM-intro*]:
$[\boldsymbol{\mathcal{A}}\varphi\ \&\ \boldsymbol{\mathcal{A}}\psi\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}(\varphi\ \&\ \psi)\ in\ v]$
unfolding *equiv-def*
by (*PLM-solver PLM-intro*: *act-conj-act-3*[*deduction*])
private lemma *ActualConjE*[*PLM-elim*,*PLM-dest*]:
$[\boldsymbol{\mathcal{A}}(\varphi\ \&\ \psi)\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi\ \&\ \boldsymbol{\mathcal{A}}\psi\ in\ v]$

unfolding *conj-def* **by** *PLM-solver*

private lemma *ActualEquivI*[*PLM-intro*]:
$[\boldsymbol{\mathcal{A}}\varphi \equiv \boldsymbol{\mathcal{A}}\psi \ in \ v] \Longrightarrow [\boldsymbol{\mathcal{A}}(\varphi \equiv \psi) \ in \ v]$
unfolding *equiv-def*
by (*PLM-solver PLM-intro*: *act-conj-act-3*[*deduction*])
private lemma *ActualEquivE*[*PLM-elim*, *PLM-dest*]:
$[\boldsymbol{\mathcal{A}}(\varphi \equiv \psi) \ in \ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi \equiv \boldsymbol{\mathcal{A}}\psi \ in \ v]$
unfolding *equiv-def* **by** *PLM-solver*

private lemma *ActualBoxI*[*PLM-intro*]:
$[\Box\varphi \ in \ v] \Longrightarrow [\boldsymbol{\mathcal{A}}(\Box\varphi) \ in \ v]$
using *qml-act-2*[*axiom-instance*, *equiv-lr*] **.**
private lemma *ActualBoxE*[*PLM-elim*, *PLM-dest*]:
$[\boldsymbol{\mathcal{A}}(\Box\varphi) \ in \ v] \Longrightarrow [\Box\varphi \ in \ v]$
using *qml-act-2*[*axiom-instance*, *equiv-rl*] **.**
private lemma *NotActualBoxD*[*PLM-dest*]:
$\neg[\boldsymbol{\mathcal{A}}(\Box\varphi) \ in \ v] \Longrightarrow \neg[\Box\varphi \ in \ v]$
using *ActualBoxI* **by** *blast*

private lemma *ActualDisjI*[*PLM-intro*]:
$[\boldsymbol{\mathcal{A}}\varphi \lor \boldsymbol{\mathcal{A}}\psi \ in \ v] \Longrightarrow [\boldsymbol{\mathcal{A}}(\varphi \lor \psi) \ in \ v]$
unfolding *disj-def* **by** *PLM-solver*
private lemma *ActualDisjE*[*PLM-elim*,*PLM-dest*]:
$[\boldsymbol{\mathcal{A}}(\varphi \lor \psi) \ in \ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi \lor \boldsymbol{\mathcal{A}}\psi \ in \ v]$
unfolding *disj-def* **by** *PLM-solver*
private lemma *NotActualDisjD*[*PLM-dest*]:
$\neg[\boldsymbol{\mathcal{A}}(\varphi \lor \psi) \ in \ v] \Longrightarrow \neg[\boldsymbol{\mathcal{A}}\varphi \lor \boldsymbol{\mathcal{A}}\psi \ in \ v]$
using *ActualDisjI* **by** *blast*

private lemma *ActualForallI*[*PLM-intro*]:
$[\forall \ x \ . \ \boldsymbol{\mathcal{A}}(\varphi \ x) \ in \ v] \Longrightarrow [\boldsymbol{\mathcal{A}}(\forall \ x \ . \ \varphi \ x) \ in \ v]$
using *logic-actual-nec-3*[*axiom-instance*, *equiv-rl*] **.**
lemma *ActualForallE*[*PLM-elim*,*PLM-dest*]:
$[\boldsymbol{\mathcal{A}}(\forall \ x \ . \ \varphi \ x) \ in \ v] \Longrightarrow [\forall \ x \ . \ \boldsymbol{\mathcal{A}}(\varphi \ x) \ in \ v]$
using *logic-actual-nec-3*[*axiom-instance*, *equiv-lr*] **.**
lemma *NotActualForallD*[*PLM-dest*]:
$\neg[\boldsymbol{\mathcal{A}}(\forall \ x \ . \ \varphi \ x) \ in \ v] \Longrightarrow \neg[\forall \ x \ . \ \boldsymbol{\mathcal{A}}(\varphi \ x) \ in \ v]$
using *ActualForallI* **by** *blast*

lemma *ActualActualI*[*PLM-intro*]:
$[\boldsymbol{\mathcal{A}}\varphi \ in \ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}\varphi \ in \ v]$
using *logic-actual-nec-4*[*axiom-instance*, *equiv-lr*] **.**
lemma *ActualActualE*[*PLM-elim*,*PLM-dest*]:
$[\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}\varphi \ in \ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi \ in \ v]$
using *logic-actual-nec-4*[*axiom-instance*, *equiv-rl*] **.**
lemma *NotActualActualD*[*PLM-dest*]:
$\neg[\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}\varphi \ in \ v] \Longrightarrow \neg[\boldsymbol{\mathcal{A}}\varphi \ in \ v]$
using *ActualActualI* **by** *blast*
end

lemma *ANeg-1*[*PLM*]:
$[\neg\boldsymbol{\mathcal{A}}\varphi \equiv \neg\varphi \ in \ dw]$
by *PLM-solver*
lemma *ANeg-2*[*PLM*]:
$[\neg\boldsymbol{\mathcal{A}}\neg\varphi \equiv \varphi \ in \ dw]$
by *PLM-solver*
lemma *Act-Basic-1*[*PLM*]:
$[\boldsymbol{\mathcal{A}}\varphi \lor \boldsymbol{\mathcal{A}}\neg\varphi \ in \ v]$
by *PLM-solver*
lemma *Act-Basic-2*[*PLM*]:
$[\boldsymbol{\mathcal{A}}(\varphi \ \& \ \psi) \equiv (\boldsymbol{\mathcal{A}}\varphi \ \& \ \boldsymbol{\mathcal{A}}\psi) \ in \ v]$
by *PLM-solver*

lemma *Act-Basic-3*[*PLM*]:
  [$\boldsymbol{\mathcal{A}}(\varphi \equiv \psi) \equiv ((\boldsymbol{\mathcal{A}}(\varphi \rightarrow \psi))$ & $(\boldsymbol{\mathcal{A}}(\psi \rightarrow \varphi)))$ *in v*]
  by *PLM-solver*
lemma *Act-Basic-4*[*PLM*]:
  [$(\boldsymbol{\mathcal{A}}(\varphi \rightarrow \psi)$ & $\boldsymbol{\mathcal{A}}(\psi \rightarrow \varphi)) \equiv (\boldsymbol{\mathcal{A}}\varphi \equiv \boldsymbol{\mathcal{A}}\psi)$ *in v*]
  by *PLM-solver*
lemma *Act-Basic-5*[*PLM*]:
  [$\boldsymbol{\mathcal{A}}(\varphi \equiv \psi) \equiv (\boldsymbol{\mathcal{A}}\varphi \equiv \boldsymbol{\mathcal{A}}\psi)$ *in v*]
  by *PLM-solver*
lemma *Act-Basic-6*[*PLM*]:
  [$\Diamond\varphi \equiv \boldsymbol{\mathcal{A}}(\Diamond\varphi)$ *in v*]
  unfolding *diamond-def* **by** *PLM-solver*
lemma *Act-Basic-7*[*PLM*]:
  [$\boldsymbol{\mathcal{A}}\varphi \equiv \Box\boldsymbol{\mathcal{A}}\varphi$ *in v*]
  by (*simp add*: *qml-2*[*axiom-instance*] *qml-act-1*[*axiom-instance*] $\equiv I$)
lemma *Act-Basic-8*[*PLM*]:
  [$\boldsymbol{\mathcal{A}}(\Box\varphi) \rightarrow \Box\boldsymbol{\mathcal{A}}\varphi$ *in v*]
  by (*metis qml-act-2*[*axiom-instance*] *CP Act-Basic-7* $\equiv E(1)$
        $\equiv E(2)$ *nec-imp-act vdash-properties-10*)
lemma *Act-Basic-9*[*PLM*]:
  [$\Box\varphi \rightarrow \Box\boldsymbol{\mathcal{A}}\varphi$ *in v*]
  using *qml-act-1*[*axiom-instance*] *ded-thm-cor-3 nec-imp-act* **by** *blast*
lemma *Act-Basic-10*[*PLM*]:
  [$\boldsymbol{\mathcal{A}}(\varphi \vee \psi) \equiv \boldsymbol{\mathcal{A}}\varphi \vee \boldsymbol{\mathcal{A}}\psi$ *in v*]
  by *PLM-solver*

lemma *Act-Basic-11*[*PLM*]:
  [$\boldsymbol{\mathcal{A}}(\exists\,\alpha.\ \varphi\ \alpha) \equiv (\exists\,\alpha.\boldsymbol{\mathcal{A}}(\varphi\ \alpha))$ *in v*]
  proof −
    have [$\boldsymbol{\mathcal{A}}(\forall\ \alpha\ .\ \neg\varphi\ \alpha) \equiv (\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha)$ *in v*]
      using *logic-actual-nec-3*[*axiom-instance*] **by** *blast*
    hence [$\neg\boldsymbol{\mathcal{A}}(\forall\ \alpha\ .\ \neg\varphi\ \alpha) \equiv \neg(\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha)$ *in v*]
      using *oth-class-taut-5-d*[*equiv-lr*] **by** *blast*
    moreover have [$\boldsymbol{\mathcal{A}}\neg(\forall\ \alpha\ .\ \neg\varphi\ \alpha) \equiv \neg\boldsymbol{\mathcal{A}}(\forall\ \alpha\ .\ \neg\varphi\ \alpha)$ *in v*]
      using *logic-actual-nec-1*[*axiom-instance*] **by** *blast*
    ultimately have [$\boldsymbol{\mathcal{A}}\neg(\forall\ \alpha\ .\ \neg\varphi\ \alpha) \equiv \neg(\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha)$ *in v*]
      using $\equiv E(5)$ **by** *auto*
    moreover {
      have [$\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha \equiv \neg\boldsymbol{\mathcal{A}}\varphi\ \alpha$ *in v*]
        using *logic-actual-nec-1*[*axiom-universal*, *axiom-instance*] **by** *blast*
      hence [$(\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha) \equiv (\forall\ \alpha\ .\ \neg\boldsymbol{\mathcal{A}}\varphi\ \alpha)$ *in v*]
        using *cqt-basic-3*[*deduction*] **by** *fast*
      hence [$(\neg(\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha)) \equiv \neg(\forall\ \alpha\ .\ \neg\boldsymbol{\mathcal{A}}\varphi\ \alpha)$ *in v*]
        using *oth-class-taut-5-d*[*equiv-lr*] **by** *blast*
    }
    ultimately show *?thesis* **unfolding** *exists-def* **using** $\equiv E(5)$ **by** *auto*
  qed

lemma *act-quant-uniq*[*PLM*]:
  [$(\forall\ z\ .\ \boldsymbol{\mathcal{A}}\varphi\ z \equiv z = x) \equiv (\forall\ z\ .\ \varphi\ z \equiv z = x)$ *in dw*]
  by *PLM-solver*

lemma *fund-cont-desc*[*PLM*]:
  [$(x^P = (\iota x.\ \varphi\ x)) \equiv (\forall\ z\ .\ \varphi\ z \equiv (z = x))$ *in dw*]
  using *descriptions*[*axiom-instance*] *act-quant-uniq* $\equiv E(5)$ **by** *fast*

lemma *hintikka*[*PLM*]:
  [$(x^P = (\iota x.\ \varphi\ x)) \equiv (\varphi\ x\ \&\ (\forall\ z\ .\ \varphi\ z \rightarrow z = x))$ *in dw*]
  proof −
    have [$(\forall\ z\ .\ \varphi\ z \equiv z = x) \equiv (\varphi\ x\ \&\ (\forall\ z\ .\ \varphi\ z \rightarrow z = x))$ *in dw*]
      unfolding *identity-ν-def* **apply** *PLM-solver* **using** *id-eq-obj-1* **apply** *simp*
      using *l-identity*[**where** $\varphi=\lambda\ x\ .\ \varphi\ x$, *axiom-instance*,
                  deduction, *deduction*]

```
      using id-eq-obj-2[deduction] unfolding identity-ν-def by fastforce
    thus ?thesis using ≡E(5) fund-cont-desc by blast
  qed

lemma russell-axiom-a[PLM]:
  [((|F, ιx. φ x|)) ≡ (∃ x . φ x & (∀ z . φ z → z = x) & (|F, x^P|)) in dw]
  (is [?lhs ≡ ?rhs in dw])
  proof −
    {
      assume 1: [?lhs in dw]
      hence [∃ α. α^P = (ιx. φ x) in dw]
      using cqt-5[axiom-instance, deduction]
            SimpleExOrEnc.intros
      by blast
      then obtain α where 2:
        [α^P = (ιx. φ x) in dw]
        using ∃E by auto
      hence 3: [φ α & (∀ z . φ z → z = α) in dw]
        using hintikka[equiv-lr] by simp
      from 2 have [(ιx. φ x) = (α^P)  in dw]
        using l-identity[where α=α^P and β=ιx. φ x and φ=λ x . x = α^P,
              axiom-instance, deduction, deduction]
              id-eq-obj-1[where x=α] by auto
      hence [(|F, α^P|) in dw]
      using 1 l-identity[where β=α^P and α=ιx. φ x and φ=λ x . (|F,x|),
                         axiom-instance, deduction, deduction] by auto
      with 3 have [φ α & (∀ z . φ z → z = α) & (|F, α^P|) in dw] by (rule &I)
      hence [?rhs in dw] using ∃I[where α=α] by simp
    }
    moreover {
      assume [?rhs in dw]
      then obtain α where 4:
        [φ α & (∀ z . φ z → z = α) & (|F, α^P|) in dw]
        using ∃E by auto
      hence [α^P = (ιx . φ x) in dw] ∧ [(|F, α^P|) in dw]
        using hintikka[equiv-rl] &E by blast
      hence [?lhs in dw]
        using l-identity[axiom-instance, deduction, deduction]
        by blast
    }
    ultimately show ?thesis by PLM-solver
  qed

lemma russell-axiom-g[PLM]:
  [{|ιx. φ x,F|} ≡ (∃ x . φ x & (∀ z . φ z → z = x) & {|x^P, F|}) in dw]
  (is [?lhs ≡ ?rhs in dw])
  proof −
    {
      assume 1: [?lhs in dw]
      hence [∃ α. α^P = (ιx. φ x) in dw]
      using cqt-5[axiom-instance, deduction] SimpleExOrEnc.intros by blast
      then obtain α where 2: [α^P = (ιx. φ x) in dw] by (rule ∃E)
      hence 3: [(φ α & (∀ z . φ z → z = α)) in dw]
        using hintikka[equiv-lr] by simp
      from 2 have [(ιx. φ x) = α^P  in dw]
        using l-identity[where α=α^P and β=ιx. φ x and φ=λ x . x = α^P,
              axiom-instance, deduction, deduction]
              id-eq-obj-1[where x=α] by auto
      hence [{|α^P, F|} in dw]
      using 1 l-identity[where β=α^P and α=ιx. φ x and φ=λ x . {|x,F|},
                         axiom-instance, deduction, deduction] by auto
      with 3 have [(φ α & (∀ z . φ z → z = α)) & {|α^P, F|} in dw]
        using &I by auto
```

hence [*?rhs in dw*] **using** $\exists I$[**where** α=α] **by** (*simp add*: *identity-defs*)
}
moreover {
assume [*?rhs in dw*]
then obtain α **where** *4*:
[$\varphi\ \alpha$ **&** ($\forall\ z\ .\ \varphi\ z \rightarrow z = \alpha$) **&** $\{\!|\alpha^P,\ F|\!\}$ *in dw*]
using $\exists E$ **by** *auto*
hence [$\alpha^P = (\iota x\ .\ \varphi\ x)$ *in dw*] $\wedge$ [$\{\!|\alpha^P,\ F|\!\}$ *in dw*]
using *hintikka*[*equiv-rl*] **&***E* **by** *blast*
hence [*?lhs in dw*]
using *l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *fast*
}
ultimately show *?thesis* **by** *PLM-solver*
qed

lemma *russell-axiom*[*PLM*]:
assumes *SimpleExOrEnc* ψ
shows [ψ ($\iota x.\ \varphi\ x$) $\equiv$ ($\exists\ x\ .\ \varphi\ x$ **&** ($\forall\ z\ .\ \varphi\ z \rightarrow z = x$) **&** ψ (x^P)) *in dw*]
(**is** [*?lhs* $\equiv$ *?rhs in dw*])
proof −
{
assume *1*: [*?lhs in dw*]
hence [$\exists\,\alpha.\ \alpha^P = (\iota x.\ \varphi\ x)$ *in dw*]
using *cqt-5*[*axiom-instance*, *deduction*] *assms* **by** *blast*
then obtain α **where** *2*: [$\alpha^P = (\iota x.\ \varphi\ x)$ *in dw*] **by** (*rule* $\exists E$)
hence *3*: [($\varphi\ \alpha$ **&** ($\forall\ z\ .\ \varphi\ z \rightarrow z = \alpha$)) *in dw*]
using *hintikka*[*equiv-lr*] **by** *simp*
from *2* **have** [($\iota x.\ \varphi\ x$) = (α^P) *in dw*]
using *l-identity*[**where** α=α^P **and** β=$\iota x.\ \varphi\ x$ **and** φ=$\lambda\ x\ .\ x = \alpha^P$,
axiom-instance, *deduction*, *deduction*]
id-eq-obj-1[**where** x=α] **by** *auto*
hence [ψ (α^P) *in dw*]
using *1* *l-identity*[**where** β=α^P **and** α=$\iota x.\ \varphi\ x$ **and** φ=$\lambda\ x\ .\ \psi\ x$,
axiom-instance, *deduction*, *deduction*] **by** *auto*
with *3* **have** [$\varphi\ \alpha$ **&** ($\forall\ z\ .\ \varphi\ z \rightarrow z = \alpha$) **&** ψ (α^P) *in dw*]
using **&***I* **by** *auto*
hence [*?rhs in dw*] **using** $\exists I$[**where** α=α] **by** (*simp add*: *identity-defs*)
}
moreover {
assume [*?rhs in dw*]
then obtain α **where** *4*:
[$\varphi\ \alpha$ **&** ($\forall\ z\ .\ \varphi\ z \rightarrow z = \alpha$) **&** ψ (α^P) *in dw*]
using $\exists E$ **by** *auto*
hence [$\alpha^P = (\iota x\ .\ \varphi\ x)$ *in dw*] $\wedge$ [ψ (α^P) *in dw*]
using *hintikka*[*equiv-rl*] **&***E* **by** *blast*
hence [*?lhs in dw*]
using *l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *fast*
}
ultimately show *?thesis* **by** *PLM-solver*
qed

lemma *unique-exists*[*PLM*]:
[($\exists\ y\ .\ y^P = (\iota x.\ \varphi\ x)$) $\equiv$ ($\exists!x\ .\ \varphi\ x$) *in dw*]
proof((*rule* $\equiv I$, *rule CP*, *rule-tac*[*2*] *CP*))
assume [$\exists\,y.\ y^P = (\iota x.\ \varphi\ x)$ *in dw*]
then obtain α **where**
[$\alpha^P = (\iota x.\ \varphi\ x)$ *in dw*]
by (*rule* $\exists E$)
hence [$\varphi\ \alpha$ **&** ($\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha$) *in dw*]
using *hintikka*[*equiv-lr*] **by** *auto*
thus [$\exists!x\ .\ \varphi\ x$ *in dw*]

unfolding *exists-unique-def* **using** $\exists I$ **by** *fast*
next
assume $[\exists !x . \varphi\ x\ in\ dw]$
then obtain α **where**
$[\varphi\ \alpha\ \&\ (\forall \beta.\ \varphi\ \beta \rightarrow \beta = \alpha)\ in\ dw]$
unfolding *exists-unique-def* **by** (*rule* $\exists E$)
hence $[\alpha^P = (\iota x.\ \varphi\ x)\ in\ dw]$
using *hintikka*[*equiv-rl*] **by** *auto*
thus $[\exists y.\ y^P = (\iota x.\ \varphi\ x)\ in\ dw]$
using $\exists I$ **by** *fast*
qed

lemma *y-in-1*[*PLM*]:
$[x^P = (\iota x . \varphi) \rightarrow \varphi\ in\ dw]$
using *hintikka*[*equiv-lr*, *conj1*] **by** (*rule CP*)

lemma *y-in-2*[*PLM*]:
$[z^P = (\iota x . \varphi\ x) \rightarrow \varphi\ z\ in\ dw]$
using *hintikka*[*equiv-lr*, *conj1*] **by** (*rule CP*)

lemma *y-in-3*[*PLM*]:
$[(\exists\ y . y^P = (\iota x . \varphi\ (x^P))) \rightarrow \varphi\ (\iota x . \varphi\ (x^P))\ in\ dw]$
proof (*rule CP*)
assume $[(\exists\ y . y^P = (\iota x . \varphi\ (x^P)))\ in\ dw]$
then obtain y **where** *1*:
$[y^P = (\iota x.\ \varphi\ (x^P))\ in\ dw]$
by (*rule* $\exists E$)
hence $[\varphi\ (y^P)\ in\ dw]$
using *y-in-2*[*deduction*] **unfolding** *identity-ν-def* **by** *blast*
thus $[\varphi\ (\iota x.\ \varphi\ (x^P))\ in\ dw]$
using *l-identity*[*axiom-instance*, *deduction*,
deduction] *1* **by** *fast*
qed

lemma *act-quant-nec*[*PLM*]:
$[(\forall z . (\mathcal{A}\varphi\ z \equiv z = x)) \equiv (\forall z.\ \mathcal{A}\mathcal{A}\varphi\ z \equiv z = x)\ in\ v]$
by *PLM-solver*

lemma *equi-desc-descA-1*[*PLM*]:
$[(x^P = (\iota x . \varphi\ x)) \equiv (x^P = (\iota x . \mathcal{A}\varphi\ x))\ in\ v]$
using *descriptions*[*axiom-instance*] **apply** (*rule* $\equiv E(5)$)
using *act-quant-nec* **apply** (*rule* $\equiv E(5)$)
using *descriptions*[*axiom-instance*]
by (*meson* $\equiv E(6)$ *oth-class-taut-4-a*)

lemma *equi-desc-descA-2*[*PLM*]:
$[(\exists\ y . y^P = (\iota x.\ \varphi\ x)) \rightarrow ((\iota x . \varphi\ x) = (\iota x . \mathcal{A}\varphi\ x))\ in\ v]$
proof (*rule CP*)
assume $[\exists y.\ y^P = (\iota x.\ \varphi\ x)\ in\ v]$
then obtain y **where**
$[y^P = (\iota x.\ \varphi\ x)\ in\ v]$
by (*rule* $\exists E$)
moreover hence $[y^P = (\iota x.\ \mathcal{A}\varphi\ x)\ in\ v]$
using *equi-desc-descA-1*[*equiv-lr*] **by** *auto*
ultimately show $[(\iota x.\ \varphi\ x) = (\iota x.\ \mathcal{A}\varphi\ x)\ in\ v]$
using *l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *fast*
qed

lemma *equi-desc-descA-3*[*PLM*]:
assumes *SimpleExOrEnc* ψ
shows $[\psi\ (\iota x.\ \varphi\ x) \rightarrow (\exists\ y . y^P = (\iota x.\ \mathcal{A}\varphi\ x))\ in\ v]$
proof (*rule CP*)

assume $[\psi\ (\iota x.\ \varphi\ x)\ in\ v]$
hence $[\exists\,\alpha.\ \alpha^P = (\iota x.\ \varphi\ x)\ in\ v]$
using *cqt-5*[*OF assms*, *axiom-instance*, *deduction*] **by** *auto*
then obtain α **where** $[\alpha^P = (\iota x.\ \varphi\ x)\ in\ v]$ **by** (*rule* $\exists\,E$)
hence $[\alpha^P = (\iota x\ .\ \boldsymbol{\mathcal{A}}\varphi\ x)\ in\ v]$
using *equi-desc-descA-1*[*equiv-lr*] **by** *auto*
thus $[\exists\,y.\ y^P = (\iota x.\ \boldsymbol{\mathcal{A}}\varphi\ x)\ in\ v]$
using $\exists\,I$ **by** *fast*
qed

lemma *equi-desc-descA-4*[*PLM*]:
assumes *SimpleExOrEnc* ψ
shows $[\psi\ (\iota x.\ \varphi\ x) \rightarrow ((\iota x.\ \varphi\ x) = (\iota x.\ \boldsymbol{\mathcal{A}}\varphi\ x))\ in\ v]$
proof (*rule CP*)
assume $[\psi\ (\iota x.\ \varphi\ x)\ in\ v]$
hence $[\exists\,\alpha.\ \alpha^P = (\iota x.\ \varphi\ x)\ in\ v]$
using *cqt-5*[*OF assms*, *axiom-instance*, *deduction*] **by** *auto*
then obtain α **where** $[\alpha^P = (\iota x.\ \varphi\ x)\ in\ v]$ **by** (*rule* $\exists\,E$)
moreover hence $[\alpha^P = (\iota x\ .\ \boldsymbol{\mathcal{A}}\varphi\ x)\ in\ v]$
using *equi-desc-descA-1*[*equiv-lr*] **by** *auto*
ultimately show $[(\iota x.\ \varphi\ x) = (\iota x\ .\ \boldsymbol{\mathcal{A}}\varphi\ x)\ in\ v]$
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
qed

lemma *nec-hintikka-scheme*[*PLM*]:
$[(x^P = (\iota x.\ \varphi\ x)) \equiv (\boldsymbol{\mathcal{A}}\varphi\ x\ \&\ (\forall\ z\ .\ \boldsymbol{\mathcal{A}}\varphi\ z \rightarrow z = x))\ in\ v]$
using *descriptions*[*axiom-instance*]
apply (*rule* $\equiv E(5)$)
apply *PLM-solver*
using *id-eq-obj-1* **apply** *simp*
using *id-eq-obj-2*[*deduction*]
l-identity[**where** $\alpha{=}x$, *axiom-instance*, *deduction*, *deduction*]
unfolding *identity-ν-def*
apply *blast*
using *l-identity*[**where** $\alpha{=}x$, *axiom-instance*, *deduction*, *deduction*]
id-eq-2[**where** $'a{=}\nu$, *deduction*] **unfolding** *identity-ν-def* **by** *meson*

lemma *equiv-desc-eq*[*PLM*]:
assumes $\bigwedge x.[\boldsymbol{\mathcal{A}}(\varphi\ x \equiv \psi\ x)\ in\ v]$
shows $[(\forall\ x\ .\ ((x^P = (\iota x\ .\ \varphi\ x)) \equiv (x^P = (\iota x\ .\ \psi\ x))))\ in\ v]$
proof(*rule* $\forall\,I$)
fix x
{
assume $[x^P = (\iota x\ .\ \varphi\ x)\ in\ v]$
hence *1*: $[\boldsymbol{\mathcal{A}}\varphi\ x\ \&\ (\forall\,z.\ \boldsymbol{\mathcal{A}}\varphi\ z \rightarrow z = x)\ in\ v]$
using *nec-hintikka-scheme*[*equiv-lr*] **by** *auto*
hence *2*: $[\boldsymbol{\mathcal{A}}\varphi\ x\ in\ v] \wedge [(\forall\,z.\ \boldsymbol{\mathcal{A}}\varphi\ z \rightarrow z = x)\ in\ v]$
using $\&E$ **by** *blast*
{
fix z
{
assume $[\boldsymbol{\mathcal{A}}\psi\ z\ in\ v]$
hence $[\boldsymbol{\mathcal{A}}\varphi\ z\ in\ v]$
using *assms*[**where** $x{=}z$] **apply** $-$ **by** *PLM-solver*
moreover have $[\boldsymbol{\mathcal{A}}\varphi\ z \rightarrow z = x\ in\ v]$
using *2* *cqt-1*[*axiom-instance*,*deduction*] **by** *auto*
ultimately have $[z = x\ in\ v]$
using *vdash-properties-10* **by** *auto*
}
hence $[\boldsymbol{\mathcal{A}}\psi\ z \rightarrow z = x\ in\ v]$ **by** (*rule CP*)
}
hence $[(\forall\ z\ .\ \boldsymbol{\mathcal{A}}\psi\ z \rightarrow z = x)\ in\ v]$ **by** (*rule* $\forall\,I$)
moreover have $[\boldsymbol{\mathcal{A}}\psi\ x\ in\ v]$

```
      using 1[conj1] assms[where x=x]
      apply − by PLM-solver
    ultimately have [𝓐ψ x & (∀ z. 𝓐ψ z → z = x) in v]
      by PLM-solver
    hence [x^P = (ιx. ψ x) in v]
     using nec-hintikka-scheme[where φ=ψ, equiv-rl] by auto
  }
  moreover {
    assume [x^P = (ιx . ψ x) in v]
    hence 1: [𝓐ψ x & (∀ z. 𝓐ψ z → z = x) in v]
      using nec-hintikka-scheme[equiv-lr] by auto
    hence 2: [𝓐ψ x in v] ∧ [(∀ z. 𝓐ψ z → z = x) in v]
      using &E by blast
    {
      fix z
      {
        assume [𝓐φ z in v]
        hence [𝓐ψ z in v]
          using assms[where x=z]
          apply − by PLM-solver
        moreover have [𝓐ψ z → z = x in v]
          using 2 cqt-1[axiom-instance,deduction] by auto
        ultimately have [z = x in v]
          using vdash-properties-10 by auto
      }
      hence [𝓐φ z → z = x in v] by (rule CP)
    }
    hence [(∀ z. 𝓐φ z → z = x) in v] by (rule ∀ I)
    moreover have [𝓐φ x in v]
      using 1[conj1] assms[where x=x]
      apply − by PLM-solver
    ultimately have [𝓐φ x & (∀ z. 𝓐φ z → z = x) in v]
      by PLM-solver
    hence [x^P = (ιx. φ x) in v]
      using nec-hintikka-scheme[where φ=φ,equiv-rl]
      by auto
  }
  ultimately show [x^P = (ιx. φ x) ≡ (x^P) = (ιx. ψ x) in v]
    using ≡I CP by auto
qed

lemma UniqueAux:
  assumes [(𝓐φ (α::ν) & (∀ z . 𝓐(φ z) → z = α)) in v]
  shows [(∀ z . (𝓐(φ z) ≡ (z = α))) in v]
  proof −
    {
      fix z
      {
        assume [𝓐(φ z) in v]
        hence [z = α in v]
          using assms[conj2, THEN cqt-1[where α=z,
                      axiom-instance, deduction],
                    deduction] by auto
      }
      moreover {
        assume [z = α in v]
        hence [α = z in v]
          unfolding identity-ν-def
          using id-eq-obj-2[deduction] by fast
        hence [𝓐(φ z) in v] using assms[conj1]
          using l-identity[axiom-instance, deduction,
                        deduction] by fast
      }
```

```
      ultimately have [(𝓐(φ z) ≡ (z = α)) in v]
        using ≡I CP by auto
    }
    thus [(∀ z . (𝓐(φ z) ≡ (z = α))) in v]
    by (rule ∀ I)
  qed

lemma nec-russell-axiom[PLM]:
  assumes SimpleExOrEnc ψ
  shows [(ψ (ιx. φ x)) ≡ (∃ x . (𝓐φ x & (∀ z . 𝓐(φ z) → z = x))
                          & ψ (x^P)) in v]
  (is [?lhs ≡ ?rhs in v])
  proof −
    {
      assume 1: [?lhs in v]
      hence [∃ α. (α^P) = (ιx. φ x) in v]
        using cqt-5[axiom-instance, deduction] assms by blast
      then obtain α where 2: [(α^P) = (ιx. φ x) in v] by (rule ∃ E)
      hence [(∀ z . (𝓐(φ z) ≡ (z = α))) in v]
        using descriptions[axiom-instance, equiv-lr] by auto
      hence 3: [(𝓐φ α) & (∀ z . (𝓐(φ z) → (z = α))) in v]
        using cqt-1[where α=α and φ=λ z . (𝓐(φ z) ≡ (z = α)),
                    axiom-instance, deduction, equiv-rl]
        using id-eq-obj-1[where x=α] unfolding identity-ν-def
        using hintikka[equiv-lr] cqt-basic-2[equiv-lr,conj1]
        &I by fast
      from 2 have [(ιx. φ x) = (α^P)  in v]
        using l-identity[where β=(ιx. φ x) and φ=λ x . x = (α^P),
              axiom-instance, deduction, deduction]
              id-eq-obj-1[where x=α] by auto
      hence [ψ (α^P) in v]
        using 1 l-identity[where α=(ιx. φ x) and φ=λ x . ψ x,
                           axiom-instance, deduction,
                           deduction] by auto
      with 3 have [(𝓐φ α & (∀ z . 𝓐(φ z) → (z = α))) & ψ (α^P) in v]
        using &I by simp
      hence [?rhs in v]
        using ∃ I[where α=α]
        by (simp add: identity-defs)
    }
    moreover {
      assume [?rhs in v]
      then obtain α where 4:
        [(𝓐φ α & (∀ z . 𝓐(φ z) → z = α)) & ψ (α^P) in v]
        using ∃ E by auto
      hence [(∀ z . (𝓐(φ z) ≡ (z = α))) in v]
        using UniqueAux &E(1) by auto
      hence [(α^P) = (ιx . φ x) in v] ∧ [ψ (α^P) in v]
        using descriptions[axiom-instance, equiv-rl]
              4[conj2] by blast
      hence [?lhs in v]
        using l-identity[axiom-instance, deduction,
                        deduction]
        by fast
    }
    ultimately show ?thesis by PLM-solver
  qed

lemma actual-desc-1[PLM]:
  [(∃ y . (y^P) = (ιx. φ x)) ≡ (∃! x . 𝓐(φ x)) in v] (is [?lhs ≡ ?rhs in v])
  proof −
    {
      assume [?lhs in v]
```

```
      then obtain α where
        [((α^P) = (ιx. φ x)) in v]
        by (rule ∃ E)
      hence [⦇A!,(ιx. φ x)⦈ in v] ∨ [(α^P) =_E (ιx. φ x) in v]
        apply − unfolding identity-defs by PLM-solver
      then obtain x where
        [((𝓐φ x & (∀ z . 𝓐(φ z) → z = x))) in v]
        using nec-russell-axiom[where ψ=λx . ⦇A!,x⦈, equiv-lr, THEN ∃ E]
        using nec-russell-axiom[where ψ=λx . (α^P) =_E x, equiv-lr, THEN ∃ E]
        using SimpleExOrEnc.intros unfolding identity_E-infix-def
        by (meson &E)
      hence [?rhs in v] unfolding exists-unique-def by (rule ∃ I)
    }
    moreover {
      assume [?rhs in v]
      then obtain x where
        [((𝓐φ x & (∀ z . 𝓐(φ z) → z = x))) in v]
        unfolding exists-unique-def by (rule ∃ E)
      hence [∀ z. 𝓐φ z ≡ z = x in v]
        using UniqueAux by auto
      hence [(x^P) = (ιx. φ x) in v]
        using descriptions[axiom-instance, equiv-rl] by auto
      hence [?lhs in v] by (rule ∃ I)
    }
    ultimately show ?thesis
      using ≡I CP by auto
  qed

lemma actual-desc-2[PLM]:
  [(x^P) = (ιx. φ) → 𝓐φ in v]
  using nec-hintikka-scheme[equiv-lr, conj1]
  by (rule CP)

lemma actual-desc-3[PLM]:
  [(z^P) = (ιx. φ x) → 𝓐(φ z) in v]
  using nec-hintikka-scheme[equiv-lr, conj1]
  by (rule CP)

lemma actual-desc-4[PLM]:
  [(∃ y . ((y^P) = (ιx. φ (x^P)))) → 𝓐(φ (ιx. φ (x^P))) in v]
  proof (rule CP)
    assume [(∃ y . (y^P) = (ιx . φ (x^P))) in v]
    then obtain y where 1:
      [y^P = (ιx. φ (x^P)) in v]
      by (rule ∃ E)
    hence [𝓐(φ (y^P)) in v] using actual-desc-3[deduction] by fast
    thus [𝓐(φ (ιx. φ (x^P))) in v]
      using l-identity[axiom-instance, deduction,
                       deduction] 1 by fast
  qed

lemma unique-box-desc-1[PLM]:
  [(∃!x . □(φ x)) → (∀ y . (y^P) = (ιx. φ x) → φ y) in v]
  proof (rule CP)
    assume [(∃!x . □(φ x)) in v]
    then obtain α where 1:
      [□φ α & (∀β. □(φ β) → β = α) in v]
      unfolding exists-unique-def by (rule ∃ E)
    {
      fix y
      {
        assume [(y^P) = (ιx. φ x) in v]
        hence [𝓐φ α → α = y in v]
```

using *nec-hintikka-scheme*[**where** $x{=}y$ **and** $\varphi{=}\varphi$, *equiv-lr*, *conj2*,
THEN *cqt-1*[**where** $\alpha{=}\alpha$,*axiom-instance*, *deduction*]] **by** *simp*
hence $[\alpha = y \ in \ v]$
using *1*[*conj1*] *nec-imp-act vdash-properties-10* **by** *blast*
hence $[\varphi \ y \ in \ v]$
using *1*[*conj1*] *qml-2*[*axiom-instance*, *deduction*]
l-identity[*axiom-instance*, *deduction*, *deduction*]
by *fast*
}
hence $[(y^P) = (\iota x. \ \varphi \ x) \rightarrow \varphi \ y \ in \ v]$
by (*rule CP*)
}
thus $[\forall \ y \ . \ (y^P) = (\iota x. \ \varphi \ x) \rightarrow \varphi \ y \ in \ v]$
by (*rule* $\forall I$)
qed

lemma *unique-box-desc*[*PLM*]:
$[(\forall \ x \ . \ (\varphi \ x \rightarrow \Box(\varphi \ x))) \rightarrow ((\exists !x \ . \ \varphi \ x)$
$\rightarrow (\forall \ y \ . \ (y^P = (\iota x \ . \ \varphi \ x)) \rightarrow \varphi \ y)) \ in \ v]$
apply (*rule CP*, *rule CP*)
using *nec-exist-unique*[*deduction*, *deduction*]
unique-box-desc-1[*deduction*] **by** *blast*

9.10 Necessity

lemma *RM-1*[*PLM*]:
$(\bigwedge v.[\varphi \rightarrow \psi \ in \ v]) \Longrightarrow [\Box\varphi \rightarrow \Box\psi \ in \ v]$
using *RN qml-1*[*axiom-instance*] *vdash-properties-10* **by** *blast*

lemma *RM-1-b*[*PLM*]:
$(\bigwedge v.[\chi \ in \ v] \Longrightarrow [\varphi \rightarrow \psi \ in \ v]) \Longrightarrow ([\Box\chi \ in \ v] \Longrightarrow [\Box\varphi \rightarrow \Box\psi \ in \ v])$
using *RN-2 qml-1*[*axiom-instance*] *vdash-properties-10* **by** *blast*

lemma *RM-2*[*PLM*]:
$(\bigwedge v.[\varphi \rightarrow \psi \ in \ v]) \Longrightarrow [\Diamond\varphi \rightarrow \Diamond\psi \ in \ v]$
unfolding *diamond-def*
using *RM-1 contraposition-1* **by** *auto*

lemma *RM-2-b*[*PLM*]:
$(\bigwedge v.[\chi \ in \ v] \Longrightarrow [\varphi \rightarrow \psi \ in \ v]) \Longrightarrow ([\Box\chi \ in \ v] \Longrightarrow [\Diamond\varphi \rightarrow \Diamond\psi \ in \ v])$
unfolding *diamond-def*
using *RM-1-b contraposition-1* **by** *blast*

lemma *KBasic-1*[*PLM*]:
$[\Box\varphi \rightarrow \Box(\psi \rightarrow \varphi) \ in \ v]$
by (*simp only*: *pl-1*[*axiom-instance*] *RM-1*)
lemma *KBasic-2*[*PLM*]:
$[\Box(\neg\varphi) \rightarrow \Box(\varphi \rightarrow \psi) \ in \ v]$
by (*simp only*: *RM-1 useful-tautologies-3*)
lemma *KBasic-3*[*PLM*]:
$[\Box(\varphi \ \& \ \psi) \equiv \Box\varphi \ \& \ \Box\psi \ in \ v]$
apply (*rule* $\equiv I$)
apply (*rule CP*)
apply (*rule* $\& I$)
using *RM-1 oth-class-taut-9-a vdash-properties-6* **apply** *blast*
using *RM-1 oth-class-taut-9-b vdash-properties-6* **apply** *blast*
using *qml-1*[*axiom-instance*] *RM-1 ded-thm-cor-3 oth-class-taut-10-a*
oth-class-taut-8-b vdash-properties-10
by *blast*
lemma *KBasic-4*[*PLM*]:
$[\Box(\varphi \equiv \psi) \equiv (\Box(\varphi \rightarrow \psi) \ \& \ \Box(\psi \rightarrow \varphi)) \ in \ v]$
apply (*rule* $\equiv I$)
unfolding *equiv-def* **using** *KBasic-3 PLM.CP* $\equiv E(1)$

apply *blast*
using *KBasic-3 PLM.CP* $\equiv E(2)$
by *blast*
lemma *KBasic-5*[*PLM*]:
$[(\Box(\varphi \rightarrow \psi)\ \&\ \Box(\psi \rightarrow \varphi)) \rightarrow (\Box\varphi \equiv \Box\psi)\ in\ v]$
by (*metis qml-1*[*axiom-instance*] *CP* **&***E* $\equiv I$ *vdash-properties-10*)
lemma *KBasic-6*[*PLM*]:
$[\Box(\varphi \equiv \psi) \rightarrow (\Box\varphi \equiv \Box\psi)\ in\ v]$
using *KBasic-4 KBasic-5* **by** (*metis equiv-def ded-thm-cor-3* **&***E*(*1*))
lemma $[(\Box\varphi \equiv \Box\psi) \rightarrow \Box(\varphi \equiv \psi)\ in\ v]$
nitpick[*expect=genuine*, *user-axioms*, *card* = *1*, *card i* = *2*]
oops — countermodel as desired
lemma *KBasic-7*[*PLM*]:
$[(\Box\varphi\ \&\ \Box\psi) \rightarrow \Box(\varphi \equiv \psi)\ in\ v]$
proof (*rule CP*)
assume $[\Box\varphi\ \&\ \Box\psi\ in\ v]$
hence $[\Box(\psi \rightarrow \varphi)\ in\ v] \wedge [\Box(\varphi \rightarrow \psi)\ in\ v]$
using **&***E KBasic-1 vdash-properties-10* **by** *blast*
thus $[\Box(\varphi \equiv \psi)\ in\ v]$
using *KBasic-4* $\equiv E(2)$ *intro-elim-1* **by** *blast*
qed

lemma *KBasic-8*[*PLM*]:
$[\Box(\varphi\ \&\ \psi) \rightarrow \Box(\varphi \equiv \psi)\ in\ v]$
using *KBasic-7 KBasic-3*
by (*metis equiv-def PLM.ded-thm-cor-3* **&***E*(*1*))
lemma *KBasic-9*[*PLM*]:
$[\Box((\neg\varphi)\ \&\ (\neg\psi)) \rightarrow \Box(\varphi \equiv \psi)\ in\ v]$
proof (*rule CP*)
assume $[\Box((\neg\varphi)\ \&\ (\neg\psi))\ in\ v]$
hence $[\Box((\neg\varphi) \equiv (\neg\psi))\ in\ v]$
using *KBasic-8 vdash-properties-10* **by** *blast*
moreover have $\bigwedge v.[((\neg\varphi) \equiv (\neg\psi)) \rightarrow (\varphi \equiv \psi)\ in\ v]$
using *CP* $\equiv E(2)$ *oth-class-taut-5-d* **by** *blast*
ultimately show $[\Box(\varphi \equiv \psi)\ in\ v]$
using *RM-1 PLM.vdash-properties-10* **by** *blast*
qed

lemma *rule-sub-lem-1-a*[*PLM*]:
$[\Box(\psi \equiv \chi)\ in\ v] \Longrightarrow [(\neg\psi) \equiv (\neg\chi)\ in\ v]$
using *qml-2*[*axiom-instance*] $\equiv E(1)$ *oth-class-taut-5-d*
vdash-properties-10
by *blast*
lemma *rule-sub-lem-1-b*[*PLM*]:
$[\Box(\psi \equiv \chi)\ in\ v] \Longrightarrow [(\psi \rightarrow \Theta) \equiv (\chi \rightarrow \Theta)\ in\ v]$
by (*metis equiv-def contraposition-1 CP* **&***E*(*2*) $\equiv I$
$\equiv E(1)$ *rule-sub-lem-1-a*)
lemma *rule-sub-lem-1-c*[*PLM*]:
$[\Box(\psi \equiv \chi)\ in\ v] \Longrightarrow [(\Theta \rightarrow \psi) \equiv (\Theta \rightarrow \chi)\ in\ v]$
by (*metis CP* $\equiv I$ $\equiv E(3)$ $\equiv E(4)$ $\neg\neg I$
$\neg\neg E$ *rule-sub-lem-1-a*)
lemma *rule-sub-lem-1-d*[*PLM*]:
$(\bigwedge x.[\Box(\psi\ x \equiv \chi\ x)\ in\ v]) \Longrightarrow [(\forall\,\alpha.\ \psi\ \alpha) \equiv (\forall\,\alpha.\ \chi\ \alpha)\ in\ v]$
by (*metis equiv-def* $\forall$ *I CP* **&***E* $\equiv I$ *raa-cor-1*
vdash-properties-10 rule-sub-lem-1-a $\forall$ *E*)
lemma *rule-sub-lem-1-e*[*PLM*]:
$[\Box(\psi \equiv \chi)\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\psi \equiv \boldsymbol{\mathcal{A}}\chi\ in\ v]$
using *Act-Basic-5* $\equiv E(1)$ *nec-imp-act*
vdash-properties-10
by *blast*
lemma *rule-sub-lem-1-f*[*PLM*]:
$[\Box(\psi \equiv \chi)\ in\ v] \Longrightarrow [\Box\psi \equiv \Box\chi\ in\ v]$
using *KBasic-6* $\equiv I$ $\equiv E(1)$ *vdash-properties-9*

by *blast*

named-theorems *Substable-intros*

definition *Substable* :: $('a \Rightarrow 'a \Rightarrow bool) \Rightarrow ('a \Rightarrow o) \Rightarrow bool$
where *Substable* $\equiv (\lambda\ cond\ \varphi\ .\ \forall\ \psi\ \chi\ v\ .\ (cond\ \psi\ \chi) \longrightarrow [\varphi\ \psi \equiv \varphi\ \chi\ in\ v])$

lemma *Substable-intro-const*[*Substable-intros*]:
Substable cond $(\lambda\ \varphi\ .\ \Theta)$
unfolding *Substable-def* **using** *oth-class-taut-4-a* **by** *blast*

lemma *Substable-intro-not*[*Substable-intros*]:
assumes *Substable cond* ψ
shows *Substable cond* $(\lambda\ \varphi\ .\ \neg(\psi\ \varphi))$
using *assms* **unfolding** *Substable-def*
using *rule-sub-lem-1-a RN-2* $\equiv E$ *oth-class-taut-5-d* **by** *metis*
lemma *Substable-intro-impl*[*Substable-intros*]:
assumes *Substable cond* ψ
and *Substable cond* χ
shows *Substable cond* $(\lambda\ \varphi\ .\ \psi\ \varphi \rightarrow \chi\ \varphi)$
using *assms* **unfolding** *Substable-def*
by (*metis* $\equiv I$ *CP intro-elim-6-a intro-elim-6-b*)
lemma *Substable-intro-box*[*Substable-intros*]:
assumes *Substable cond* ψ
shows *Substable cond* $(\lambda\ \varphi\ .\ \Box(\psi\ \varphi))$
using *assms* **unfolding** *Substable-def*
using *rule-sub-lem-1-f RN* **by** *meson*
lemma *Substable-intro-actual*[*Substable-intros*]:
assumes *Substable cond* ψ
shows *Substable cond* $(\lambda\ \varphi\ .\ \boldsymbol{\mathcal{A}}(\psi\ \varphi))$
using *assms* **unfolding** *Substable-def*
using *rule-sub-lem-1-e RN* **by** *meson*
lemma *Substable-intro-all*[*Substable-intros*]:
assumes $\forall\ x\ .$ *Substable cond* $(\psi\ x)$
shows *Substable cond* $(\lambda\ \varphi\ .\ \forall\ x\ .\ \psi\ x\ \varphi)$
using *assms* **unfolding** *Substable-def*
by (*simp add*: *RN rule-sub-lem-1-d*)

named-theorems *Substable-Cond-defs*
end

class *Substable* =
fixes *Substable-Cond* :: $'a \Rightarrow 'a \Rightarrow bool$
assumes *rule-sub-nec*:
$\bigwedge\ \varphi\ \psi\ \chi\ \Theta\ v\ .\ [\![PLM.Substable\ Substable\text{-}Cond\ \varphi;\ Substable\text{-}Cond\ \psi\ \chi]\!]$
$\Longrightarrow \Theta\ [\varphi\ \psi\ in\ v] \Longrightarrow \Theta\ [\varphi\ \chi\ in\ v]$

instantiation o :: *Substable*
begin
definition *Substable-Cond*-o **where** [*PLM.Substable-Cond-defs*]:
Substable-Cond-o $\equiv \lambda\ \varphi\ \psi\ .\ \forall\ v\ .\ [\varphi \equiv \psi\ in\ v]$
instance proof
interpret *PLM* **.**
fix φ :: o $\Rightarrow$ o **and** $\psi\ \chi$ :: o **and** Θ :: *bool* $\Rightarrow$ *bool* **and** *v*::*i*
assume *Substable Substable-Cond* φ
moreover assume *Substable-Cond* $\psi\ \chi$
ultimately have $[\varphi\ \psi \equiv \varphi\ \chi\ in\ v]$
unfolding *Substable-def* **by** *blast*
hence $[\varphi\ \psi\ in\ v] = [\varphi\ \chi\ in\ v]$ **using** $\equiv E$ **by** *blast*
moreover assume $\Theta\ [\varphi\ \psi\ in\ v]$
ultimately show $\Theta\ [\varphi\ \chi\ in\ v]$ **by** *simp*
qed

end

instantiation *fun* :: (*type*, *Substable*) *Substable*
begin
  definition *Substable-Cond-fun* **where** [*PLM.Substable-Cond-defs*]:
    Substable-Cond-fun ≡ λ φ ψ . ∀ x . *Substable-Cond* (φ x) (ψ x)
  instance proof
    interpret *PLM* **.**
    fix φ:: ('a ⇒ 'b) ⇒ o **and** ψ χ :: 'a ⇒ 'b **and** Θ v
    assume *Substable Substable-Cond* φ
    moreover assume *Substable-Cond* ψ χ
    ultimately have [φ ψ ≡ φ χ *in* v]
      unfolding *Substable-def* **by** *blast*
    hence [φ ψ *in* v] = [φ χ *in* v] **using** ≡*E* **by** *blast*
    moreover assume Θ [φ ψ *in* v]
    ultimately show Θ [φ χ *in* v] **by** *simp*
  qed
end

context *PLM*
begin

  lemma *Substable-intro-equiv*[*Substable-intros*]:
    assumes *Substable cond* ψ
      and *Substable cond* χ
    shows *Substable cond* (λ φ . ψ φ ≡ χ φ)
    unfolding *conn-defs* **by** (*simp add*: *assms Substable-intros*)
  lemma *Substable-intro-conj*[*Substable-intros*]:
    assumes *Substable cond* ψ
      and *Substable cond* χ
    shows *Substable cond* (λ φ . ψ φ **&** χ φ)
    unfolding *conn-defs* **by** (*simp add*: *assms Substable-intros*)
  lemma *Substable-intro-disj*[*Substable-intros*]:
    assumes *Substable cond* ψ
      and *Substable cond* χ
    shows *Substable cond* (λ φ . ψ φ ∨ χ φ)
    unfolding *conn-defs* **by** (*simp add*: *assms Substable-intros*)
  lemma *Substable-intro-diamond*[*Substable-intros*]:
    assumes *Substable cond* ψ
    shows *Substable cond* (λ φ . ◊(ψ φ))
    unfolding *conn-defs* **by** (*simp add*: *assms Substable-intros*)
  lemma *Substable-intro-exist*[*Substable-intros*]:
    assumes ∀ x . *Substable cond* (ψ x)
    shows *Substable cond* (λ φ . ∃ x . ψ x φ)
    unfolding *conn-defs* **by** (*simp add*: *assms Substable-intros*)

  lemma *Substable-intro-id-*o[*Substable-intros*]:
    Substable Substable-Cond (λ φ . φ)
    unfolding *Substable-def Substable-Cond-*o*-def* **by** *blast*
  lemma *Substable-intro-id-fun*[*Substable-intros*]:
    assumes *Substable Substable-Cond* ψ
    shows *Substable Substable-Cond* (λ φ . ψ (φ x))
    using *assms* **unfolding** *Substable-def Substable-Cond-fun-def*
    by *blast*

  method *PLM-subst-method* **for** ψ::'a::*Substable* **and** χ::'a::*Substable* =
    (*match* **conclusion in** Θ [φ χ *in* v] **for** Θ **and** φ **and** v ⇒
      ‹(*rule rule-sub-nec*[*where* Θ=Θ *and* χ=χ *and* ψ=ψ *and* φ=φ *and* v=v],
        ((*fast intro*: *Substable-intros*, ((*assumption*)+)?)+; *fail*),
        unfold Substable-Cond-defs)›)

  method *PLM-autosubst* =
    (*match* **premises in** ⋀v . [ψ ≡ χ *in* v] **for** ψ **and** χ ⇒

⟨ *match conclusion in* Θ [φ χ *in v*] *for* Θ φ *and v* ⇒
⟨(*rule rule-sub-nec*[*where* Θ=Θ *and* χ=χ *and* ψ=ψ *and* φ=φ *and v*=*v*],
((*fast intro*: *Substable-intros*, ((*assumption*)+)?)+; *fail*),
unfold Substable-Cond-defs)⟩ ⟩)

method *PLM-autosubst1* =
(*match* **premises in** $\bigwedge v\ x$. [ψ x $\equiv$ χ x *in v*]
for ψ::'*a*::*type*⇒o **and** χ::'*a*⇒o ⇒
⟨ *match conclusion in* Θ [φ χ *in v*] *for* Θ φ *and v* ⇒
⟨(*rule rule-sub-nec*[*where* Θ=Θ *and* χ=χ *and* ψ=ψ *and* φ=φ *and v*=*v*],
((*fast intro*: *Substable-intros*, ((*assumption*)+)?)+; *fail*),
unfold Substable-Cond-defs)⟩ ⟩)

method *PLM-autosubst2* =
(*match* **premises in** $\bigwedge v\ x\ y$. [ψ x y $\equiv$ χ x y *in v*]
for ψ::'*a*::*type*⇒'*a*⇒o **and** χ::'*a*::*type*⇒'*a*⇒o ⇒
⟨ *match conclusion in* Θ [φ χ *in v*] *for* Θ φ *and v* ⇒
⟨(*rule rule-sub-nec*[*where* Θ=Θ *and* χ=χ *and* ψ=ψ *and* φ=φ *and v*=*v*],
((*fast intro*: *Substable-intros*, ((*assumption*)+)?)+; *fail*),
unfold Substable-Cond-defs)⟩ ⟩)

method *PLM-subst-goal-method* **for** φ::'*a*::*Substable*⇒o **and** ψ::'*a* =
(*match* **conclusion in** Θ [φ χ *in v*] **for** Θ **and** χ **and** *v* ⇒
⟨(*rule rule-sub-nec*[*where* Θ=Θ *and* χ=χ *and* ψ=ψ *and* φ=φ *and v*=*v*],
((*fast intro*: *Substable-intros*, ((*assumption*)+)?)+; *fail*),
unfold Substable-Cond-defs)⟩)

lemma *rule-sub-nec*[*PLM*]:
assumes *Substable Substable-Cond* φ
shows ($\bigwedge v$.[(ψ $\equiv$ χ) *in v*]) $\Longrightarrow$ Θ [φ ψ *in v*] $\Longrightarrow$ Θ [φ χ *in v*]
proof −
assume ($\bigwedge v$.[(ψ $\equiv$ χ) *in v*])
hence [φ ψ *in v*] = [φ χ *in v*]
using *assms RN* **unfolding** *Substable-def Substable-Cond-defs*
using ≡*I CP* ≡*E*(*1*) ≡*E*(*2*) **by** *meson*
thus Θ [φ ψ *in v*] $\Longrightarrow$ Θ [φ χ *in v*] **by** *auto*
qed

lemma *rule-sub-nec1*[*PLM*]:
assumes *Substable Substable-Cond* φ
shows ($\bigwedge v\ x$.[(ψ x $\equiv$ χ x) *in v*]) $\Longrightarrow$ Θ [φ ψ *in v*] $\Longrightarrow$ Θ [φ χ *in v*]
proof −
assume ($\bigwedge v\ x$.[(ψ x $\equiv$ χ x) *in v*])
hence [φ ψ *in v*] = [φ χ *in v*]
using *assms RN* **unfolding** *Substable-def Substable-Cond-defs*
using ≡*I CP* ≡*E*(*1*) ≡*E*(*2*) **by** *metis*
thus Θ [φ ψ *in v*] $\Longrightarrow$ Θ [φ χ *in v*] **by** *auto*
qed

lemma *rule-sub-nec2*[*PLM*]:
assumes *Substable Substable-Cond* φ
shows ($\bigwedge v\ x\ y$.[ψ x y $\equiv$ χ x y *in v*]) $\Longrightarrow$ Θ [φ ψ *in v*] $\Longrightarrow$ Θ [φ χ *in v*]
proof −
assume ($\bigwedge v\ x\ y$.[ψ x y $\equiv$ χ x y *in v*])
hence [φ ψ *in v*] = [φ χ *in v*]
using *assms RN* **unfolding** *Substable-def Substable-Cond-defs*
using ≡*I CP* ≡*E*(*1*) ≡*E*(*2*) **by** *metis*
thus Θ [φ ψ *in v*] $\Longrightarrow$ Θ [φ χ *in v*] **by** *auto*
qed

lemma *rule-sub-remark-1-autosubst*:

assumes $(\bigwedge v.[(\!|A!,x|\!) \equiv (\neg(\Diamond(\!|E!,x|\!)))\ \mathit{in}\ v])$
and $[\neg(\!|A!,x|\!)\ \mathit{in}\ v]$
shows$[\neg\neg\Diamond(\!|E!,x|\!)\ \mathit{in}\ v]$
apply (*insert assms*) **apply** *PLM-autosubst* **by** *auto*

lemma *rule-sub-remark-1*:
assumes $(\bigwedge v.[(\!|A!,x|\!) \equiv (\neg(\Diamond(\!|E!,x|\!)))\ \mathit{in}\ v])$
and $[\neg(\!|A!,x|\!)\ \mathit{in}\ v]$
shows$[\neg\neg\Diamond(\!|E!,x|\!)\ \mathit{in}\ v]$
apply (*PLM-subst-method* $(\!|A!,x|\!)$ $(\neg(\Diamond(\!|E!,x|\!)))$)
apply (*simp add*: *assms*(*1*))
by (*simp add*: *assms*(*2*))

lemma *rule-sub-remark-2*:
assumes $(\bigwedge v.[(\!|R,x,y|\!) \equiv ((\!|R,x,y|\!)\ \&\ ((\!|Q,a|\!) \vee (\neg(\!|Q,a|\!))))\ \mathit{in}\ v])$
and $[p \to (\!|R,x,y|\!)\ \mathit{in}\ v]$
shows$[p \to ((\!|R,x,y|\!)\ \&\ ((\!|Q,a|\!) \vee (\neg(\!|Q,a|\!))))\ \ \mathit{in}\ v]$
apply (*insert assms*) **apply** *PLM-autosubst* **by** *auto*

lemma *rule-sub-remark-3-autosubst*:
assumes $(\bigwedge v\ x.[(\!|A!,x^P|\!) \equiv (\neg(\Diamond(\!|E!,x^P|\!)))\ \mathit{in}\ v])$
and $[\exists\ \ x\ .\ (\!|A!,x^P|\!)\ \mathit{in}\ v]$
shows$[\exists\ \ x\ .\ (\neg(\Diamond(\!|E!,x^P|\!)))\ \ \mathit{in}\ v]$
apply (*insert assms*) **apply** *PLM-autosubst1* **by** *auto*

lemma *rule-sub-remark-3*:
assumes $(\bigwedge v\ x.[(\!|A!,x^P|\!) \equiv (\neg(\Diamond(\!|E!,x^P|\!)))\ \mathit{in}\ v])$
and $[\exists\ \ x\ .\ (\!|A!,x^P|\!)\ \mathit{in}\ v]$
shows $[\exists\ \ x\ .\ (\neg(\Diamond(\!|E!,x^P|\!)))\ \ \mathit{in}\ v]$
apply (*PLM-subst-method* $\lambda x\ .\ (\!|A!,x^P|\!)$ $\lambda x\ .\ (\neg(\Diamond(\!|E!,x^P|\!)))$)
apply (*simp add*: *assms*(*1*))
by (*simp add*: *assms*(*2*))

lemma *rule-sub-remark-4*:
assumes $\bigwedge v\ x.[(\neg(\neg(\!|P,x^P|\!))) \equiv (\!|P,x^P|\!)\ \mathit{in}\ v]$
and $[\boldsymbol{\mathcal{A}}(\neg(\neg(\!|P,x^P|\!)))\ \mathit{in}\ v]$
shows $[\boldsymbol{\mathcal{A}}(\!|P,x^P|\!)\ \mathit{in}\ v]$
apply (*insert assms*) **apply** *PLM-autosubst1* **by** *auto*

lemma *rule-sub-remark-5*:
assumes $\bigwedge v.[(\varphi \to \psi) \equiv ((\neg\psi) \to (\neg\varphi))\ \mathit{in}\ v]$
and $[\Box(\varphi \to \psi)\ \mathit{in}\ v]$
shows $[\Box((\neg\psi) \to (\neg\varphi))\ \mathit{in}\ v]$
apply (*insert assms*) **apply** *PLM-autosubst* **by** *auto*

lemma *rule-sub-remark-6*:
assumes $\bigwedge v.[\psi \equiv \chi\ \mathit{in}\ v]$
and $[\Box(\varphi \to \psi)\ \mathit{in}\ v]$
shows $[\Box(\varphi \to \chi)\ \mathit{in}\ v]$
apply (*insert assms*) **apply** *PLM-autosubst* **by** *auto*

lemma *rule-sub-remark-7*:
assumes $\bigwedge v.[\varphi \equiv (\neg(\neg\varphi))\ \mathit{in}\ v]$
and $[\Box(\varphi \to \varphi)\ \mathit{in}\ v]$
shows $[\Box((\neg(\neg\varphi)) \to \varphi)\ \mathit{in}\ v]$
apply (*insert assms*) **apply** *PLM-autosubst* **by** *auto*

lemma *rule-sub-remark-8*:
assumes $\bigwedge v.[\boldsymbol{\mathcal{A}}\varphi \equiv \varphi\ \mathit{in}\ v]$
and $[\Box(\boldsymbol{\mathcal{A}}\varphi)\ \mathit{in}\ v]$
shows $[\Box(\varphi)\ \mathit{in}\ v]$
apply (*insert assms*) **apply** *PLM-autosubst* **by** *auto*

lemma *rule-sub-remark-9*:
  assumes $\bigwedge v.[(\!|P,a|\!) \equiv ((\!|P,a|\!)\ \&\ ((\!|Q,b|\!) \lor (\neg(\!|Q,b|\!)))) \ in\ v]$
    and $[(\!|P,a|\!) = (\!|P,a|\!) \ in\ v]$
  shows $[(\!|P,a|\!) = ((\!|P,a|\!)\ \&\ ((\!|Q,b|\!) \lor (\neg(\!|Q,b|\!)))) \ in\ v]$
   unfolding *identity-defs* **apply** (*insert assms*)
   apply *PLM-autosubst* **oops** — no match as desired

— *dr-alphabetic-rules* implicitly holds
— *dr-alphabetic-thm* implicitly holds

lemma *KBasic2-1*[*PLM*]:
  $[\Box\varphi \equiv \Box(\neg(\neg\varphi)) \ in\ v]$
  apply (*PLM-subst-method* φ $(\neg(\neg\varphi))$)
   by *PLM-solver+*

lemma *KBasic2-2*[*PLM*]:
  $[(\neg(\Box\varphi)) \equiv \Diamond(\neg\varphi) \ in\ v]$
  unfolding *diamond-def*
  apply (*PLM-subst-method* φ $\neg(\neg\varphi)$)
   by *PLM-solver+*

lemma *KBasic2-3*[*PLM*]:
  $[\Box\varphi \equiv (\neg(\Diamond(\neg\varphi))) \ in\ v]$
  unfolding *diamond-def*
  apply (*PLM-subst-method* φ $\neg(\neg\varphi)$)
   apply *PLM-solver*
  by (*simp add*: *oth-class-taut-4-b*)
lemmas *Df*$\Box$ = *KBasic2-3*

lemma *KBasic2-4*[*PLM*]:
  $[\Box(\neg(\varphi)) \equiv (\neg(\Diamond\varphi)) \ in\ v]$
  unfolding *diamond-def*
  by (*simp add*: *oth-class-taut-4-b*)

lemma *KBasic2-5*[*PLM*]:
  $[\Box(\varphi \rightarrow \psi) \rightarrow (\Diamond\varphi \rightarrow \Diamond\psi) \ in\ v]$
  by (*simp only*: *CP RM-2-b*)
lemmas *K*$\Diamond$ = *KBasic2-5*

lemma *KBasic2-6*[*PLM*]:
  $[\Diamond(\varphi \lor \psi) \equiv (\Diamond\varphi \lor \Diamond\psi) \ in\ v]$
  proof –
   have $[\Box((\neg\varphi)\ \&\ (\neg\psi)) \equiv (\Box(\neg\varphi)\ \&\ \Box(\neg\psi)) \ in\ v]$
    using *KBasic-3* **by** *blast*
   hence $[(\neg(\Diamond(\neg((\neg\varphi)\ \&\ (\neg\psi))))) \equiv (\Box(\neg\varphi)\ \&\ \Box(\neg\psi)) \ in\ v]$
    using *Df*$\Box$ **by** (*rule* $\equiv E(6)$)
   hence $[(\neg(\Diamond(\neg((\neg\varphi)\ \&\ (\neg\psi))))) \equiv ((\neg(\Diamond\varphi))\ \&\ (\neg(\Diamond\psi))) \ in\ v]$
    apply – **apply** (*PLM-subst-method* $\Box(\neg\varphi)$ $\neg(\Diamond\varphi)$)
     apply (*simp add*: *KBasic2-4*)
    apply (*PLM-subst-method* $\Box(\neg\psi)$ $\neg(\Diamond\psi)$)
     apply (*simp add*: *KBasic2-4*)
    unfolding *diamond-def* **by** *assumption*
   hence $[(\neg(\Diamond(\varphi \lor \psi))) \equiv ((\neg(\Diamond\varphi))\ \&\ (\neg(\Diamond\psi))) \ in\ v]$
    apply – **apply** (*PLM-subst-method* $\neg((\neg\varphi)\ \&\ (\neg\psi))$ $\varphi \lor \psi$)
    using *oth-class-taut-6-b*[*equiv-sym*] **by** *auto*
   hence $[(\neg(\neg(\Diamond(\varphi \lor \psi)))) \equiv (\neg((\neg(\Diamond\varphi))\&(\neg(\Diamond\psi)))) \ in\ v]$
    by (*rule oth-class-taut-5-d*[*equiv-lr*])
   hence $[\Diamond(\varphi \lor \psi) \equiv (\neg((\neg(\Diamond\varphi))\ \&\ (\neg(\Diamond\psi)))) \ in\ v]$
    apply – **apply** (*PLM-subst-method* $\neg(\neg(\Diamond(\varphi \lor \psi)))$ $\Diamond(\varphi \lor \psi)$)
    using *oth-class-taut-4-b*[*equiv-sym*] **by** *auto*
   thus *?thesis*
    apply – **apply** (*PLM-subst-method* $\neg((\neg(\Diamond\varphi))\ \&\ (\neg(\Diamond\psi)))$ $(\Diamond\varphi) \lor (\Diamond\psi)$)
    using *oth-class-taut-6-b*[*equiv-sym*] **by** *auto*

```
  qed

lemma KBasic2-7[PLM]:
  [(□φ ∨ □ψ) → □(φ ∨ ψ) in v]
  proof −
    have ⋀ v . [φ → (φ ∨ ψ) in v]
      by (metis contraposition-1 contraposition-2 useful-tautologies-3 disj-def)
    hence [□φ → □(φ ∨ ψ) in v] using RM-1 by auto
    moreover {
        have ⋀ v . [ψ → (φ ∨ ψ) in v]
          by (simp only: pl-1[axiom-instance] disj-def)
        hence [□ψ → □(φ ∨ ψ) in v]
          using RM-1 by auto
    }
    ultimately show ?thesis
      using oth-class-taut-10-d vdash-properties-10 by blast
  qed

lemma KBasic2-8[PLM]:
  [◇(φ & ψ) → (◇φ & ◇ψ) in v]
  by (metis CP RM-2 &I oth-class-taut-9-a
          oth-class-taut-9-b vdash-properties-10)

lemma KBasic2-9[PLM]:
  [◇(φ → ψ) ≡ (□φ → ◇ψ) in v]
  apply (PLM-subst-method (¬(□φ)) ∨ (◇ψ) □φ → ◇ψ)
   using oth-class-taut-5-k[equiv-sym] apply simp
  apply (PLM-subst-method (¬φ) ∨ ψ φ → ψ)
   using oth-class-taut-5-k[equiv-sym] apply simp
  apply (PLM-subst-method ◇(¬φ) ¬(□φ))
   using KBasic2-2[equiv-sym] apply simp
  using KBasic2-6 .

lemma KBasic2-10[PLM]:
  [◇(□φ) ≡ (¬(□◇(¬φ))) in v]
  unfolding diamond-def apply (PLM-subst-method φ ¬¬φ)
  using oth-class-taut-4-b oth-class-taut-4-a by auto

lemma KBasic2-11[PLM]:
  [◇◇φ ≡ (¬(□□(¬φ))) in v]
  unfolding diamond-def
  apply (PLM-subst-method □(¬φ) ¬(¬(□(¬φ))))
  using oth-class-taut-4-b oth-class-taut-4-a by auto

lemma KBasic2-12[PLM]: [□(φ ∨ ψ) → (□φ ∨ ◇ψ) in v]
  proof −
    have [□(ψ ∨ φ) → (□(¬ψ) → □φ) in v]
      using CP RM-1-b ∨E(2) by blast
    hence [□(ψ ∨ φ) → (◇ψ ∨ □φ) in v]
      unfolding diamond-def disj-def
      by (meson CP ¬¬E vdash-properties-6)
    thus ?thesis apply −
      apply (PLM-subst-method (◇ψ ∨ □φ) (□φ ∨ ◇ψ))
       apply (simp add: PLM.oth-class-taut-3-e)
      apply (PLM-subst-method (ψ ∨ φ) (φ ∨ ψ))
       apply (simp add: PLM.oth-class-taut-3-e)
      by assumption
  qed

lemma TBasic[PLM]:
  [φ → ◇φ in v]
  unfolding diamond-def
  apply (subst contraposition-1)
```

```
  apply (PLM-subst-method □¬φ ¬¬□¬φ)
   apply (simp add: PLM.oth-class-taut-4-b)
  using qml-2[where φ=¬φ, axiom-instance]
  by simp
lemmas T◊ = TBasic

lemma S5Basic-1[PLM]:
  [◊□φ → □φ in v]
  proof (rule CP)
    assume [◊□φ in v]
    hence [¬□◊¬φ in v]
      using KBasic2-10[equiv-lr] by simp
    moreover have [◊(¬φ) → □◊(¬φ) in v]
      by (simp add: qml-3[axiom-instance])
    ultimately have [¬◊¬φ in v]
      by (simp add: PLM.modus-tollens-1)
    thus [□φ in v]
      unfolding diamond-def apply −
      apply (PLM-subst-method ¬¬φ φ)
       using oth-class-taut-4-b[equiv-sym] apply simp
      unfolding diamond-def using oth-class-taut-4-b[equiv-rl]
      by simp
  qed
lemmas 5◊ = S5Basic-1

lemma S5Basic-2[PLM]:
  [□φ ≡ ◊□φ in v]
  using 5◊ T◊ ≡I by blast

lemma S5Basic-3[PLM]:
  [◊φ ≡ □◊φ in v]
  using qml-3[axiom-instance] qml-2[axiom-instance] ≡I by blast

lemma S5Basic-4[PLM]:
  [φ → □◊φ in v]
  using T◊[deduction, THEN S5Basic-3[equiv-lr]]
  by (rule CP)

lemma S5Basic-5[PLM]:
  [◊□φ → φ in v]
  using S5Basic-2[equiv-rl, THEN qml-2[axiom-instance, deduction]]
  by (rule CP)
lemmas B◊ = S5Basic-5

lemma S5Basic-6[PLM]:
  [□φ → □□φ in v]
  using S5Basic-4[deduction] RM-1[OF S5Basic-1, deduction] CP by auto
lemmas 4□ = S5Basic-6

lemma S5Basic-7[PLM]:
  [□φ ≡ □□φ in v]
  using 4□ qml-2[axiom-instance] by (rule ≡I)

lemma S5Basic-8[PLM]:
  [◊◊φ → ◊φ in v]
  using S5Basic-6[where φ=¬φ, THEN contraposition-1[THEN iffD1], deduction]
        KBasic2-11[equiv-lr] CP unfolding diamond-def by auto
lemmas 4◊ = S5Basic-8

lemma S5Basic-9[PLM]:
  [◊◊φ ≡ ◊φ in v]
  using 4◊ T◊ by (rule ≡I)
```

```
lemma S5Basic-10[PLM]:
  [□(φ ∨ □ψ) ≡ (□φ ∨ □ψ) in v]
  apply (rule ≡I)
   apply (PLM-subst-goal-method λ χ . □(φ ∨ □ψ) → (□φ ∨ χ) ◇□ψ)
    using S5Basic-2[equiv-sym] apply simp
   using KBasic2-12 apply assumption
  apply (PLM-subst-goal-method λ χ .(□φ ∨ χ) → □(φ ∨ □ψ) □□ψ)
   using S5Basic-7[equiv-sym] apply simp
  using KBasic2-7 by auto

lemma S5Basic-11[PLM]:
  [□(φ ∨ ◇ψ) ≡ (□φ ∨ ◇ψ) in v]
  apply (rule ≡I)
   apply (PLM-subst-goal-method λ χ . □(φ ∨ ◇ψ) → (□φ ∨ χ) ◇◇ψ)
    using S5Basic-9 apply simp
   using KBasic2-12 apply assumption
  apply (PLM-subst-goal-method λ χ .(□φ ∨ χ) → □(φ ∨ ◇ψ) □◇ψ)
   using S5Basic-3[equiv-sym] apply simp
  using KBasic2-7 by assumption

lemma S5Basic-12[PLM]:
  [◇(φ & ◇ψ) ≡ (◇φ & ◇ψ) in v]
  proof −
    have [□((¬φ) ∨ □(¬ψ)) ≡ (□(¬φ) ∨ □(¬ψ)) in v]
      using S5Basic-10 by auto
    hence 1: [(¬□((¬φ) ∨ □(¬ψ))) ≡ ¬(□(¬φ) ∨ □(¬ψ)) in v]
      using oth-class-taut-5-d[equiv-lr] by auto
    have 2: [(◇(¬((¬φ) ∨ (¬(◇ψ))))) ≡ (¬((¬(◇φ)) ∨ (¬(◇ψ)))) in v]
      apply (PLM-subst-method □¬ψ ¬◇ψ)
       using KBasic2-4 apply simp
      apply (PLM-subst-method □¬φ ¬◇φ)
       using KBasic2-4 apply simp
      apply (PLM-subst-method (¬□((¬φ) ∨ □(¬ψ))) (◇(¬((¬φ) ∨ (□(¬ψ))))))
       unfolding diamond-def
       apply (simp add: RN oth-class-taut-4-b rule-sub-lem-1-a rule-sub-lem-1-f)
      using 1 by assumption
    show ?thesis
      apply (PLM-subst-method ¬((¬φ) ∨ (¬◇ψ)) φ & ◇ψ)
       using oth-class-taut-6-a[equiv-sym] apply simp
      apply (PLM-subst-method ¬((¬(◇φ)) ∨ (¬◇ψ)) ◇φ & ◇ψ)
       using oth-class-taut-6-a[equiv-sym] apply simp
      using 2 by assumption
  qed

lemma S5Basic-13[PLM]:
  [◇(φ & (□ψ)) ≡ (◇φ & (□ψ)) in v]
  apply (PLM-subst-method ◇□ψ □ψ)
   using S5Basic-2[equiv-sym] apply simp
  using S5Basic-12 by simp

lemma S5Basic-14[PLM]:
  [□(φ → (□ψ)) ≡ □(◇φ → ψ) in v]
  proof (rule ≡I; rule CP)
    assume [□(φ → □ψ) in v]
    moreover {
      have ⋀v.[□(φ → □ψ) → (◇φ → ψ) in v]
        proof (rule CP)
          fix v
          assume [□(φ → □ψ) in v]
          hence [◇φ → ◇□ψ in v]
            using K◇[deduction] by auto
          thus [◇φ → ψ in v]
            using B◇ ded-thm-cor-3 by blast
```

qed
hence $[\Box(\Box(\varphi \to \Box\psi) \to (\Diamond\varphi \to \psi))$ *in* $v]$
by (*rule RN*)
hence $[\Box(\Box(\varphi \to \Box\psi)) \to \Box((\Diamond\varphi \to \psi))$ *in* $v]$
using *qml-1*[*axiom-instance*, *deduction*] **by** *auto*
}
ultimately show $[\Box(\Diamond\varphi \to \psi)$ *in* $v]$
using *S5Basic-6 CP vdash-properties-10* **by** *meson*
next
assume $[\Box(\Diamond\varphi \to \psi)$ *in* $v]$
moreover {
fix v
{
assume $[\Box(\Diamond\varphi \to \psi)$ *in* $v]$
hence *1*: $[\Box\Diamond\varphi \to \Box\psi$ *in* $v]$
using *qml-1*[*axiom-instance*, *deduction*] **by** *auto*
assume $[\varphi$ *in* $v]$
hence $[\Box\Diamond\varphi$ *in* $v]$
using *S5Basic-4*[*deduction*] **by** *auto*
hence $[\Box\psi$ *in* $v]$
using *1*[*deduction*] **by** *auto*
}
hence $[\Box(\Diamond\varphi \to \psi)$ *in* $v] \Longrightarrow [\varphi \to \Box\psi$ *in* $v]$
using *CP* **by** *auto*
}
ultimately show $[\Box(\varphi \to \Box\psi)$ *in* $v]$
using *S5Basic-6 RN-2 vdash-properties-10* **by** *blast*
qed

lemma *sc-eq-box-box-1*[*PLM*]:
$[\Box(\varphi \to \Box\varphi) \to (\Diamond\varphi \equiv \Box\varphi)$ *in* $v]$
proof(*rule CP*)
assume *1*: $[\Box(\varphi \to \Box\varphi)$ *in* $v]$
hence $[\Box(\Diamond\varphi \to \varphi)$ *in* $v]$
using *S5Basic-14*[*equiv-lr*] **by** *auto*
hence $[\Diamond\varphi \to \varphi$ *in* $v]$
using *qml-2*[*axiom-instance*, *deduction*] **by** *auto*
moreover from *1* **have** $[\varphi \to \Box\varphi$ *in* $v]$
using *qml-2*[*axiom-instance*, *deduction*] **by** *auto*
ultimately have $[\Diamond\varphi \to \Box\varphi$ *in* $v]$
using *ded-thm-cor-3* **by** *auto*
moreover have $[\Box\varphi \to \Diamond\varphi$ *in* $v]$
using *qml-2*[*axiom-instance*] $T\Diamond$
by (*rule ded-thm-cor-3*)
ultimately show $[\Diamond\varphi \equiv \Box\varphi$ *in* $v]$
by (*rule* $\equiv I$)
qed

lemma *sc-eq-box-box-2*[*PLM*]:
$[\Box(\varphi \to \Box\varphi) \to ((\neg\Box\varphi) \equiv (\Box(\neg\varphi)))$ *in* $v]$
proof (*rule CP*)
assume $[\Box(\varphi \to \Box\varphi)$ *in* $v]$
hence $[(\neg\Box(\neg\varphi)) \equiv \Box\varphi$ *in* $v]$
using *sc-eq-box-box-1*[*deduction*] **unfolding** *diamond-def* **by** *auto*
thus $[((\neg\Box\varphi) \equiv (\Box(\neg\varphi)))$ *in* $v]$
by (*meson CP* $\equiv I$ $\equiv E(3)$
$\equiv E(4)$ $\neg\neg I$ $\neg\neg E$)
qed

lemma *sc-eq-box-box-3*[*PLM*]:
$[(\Box(\varphi \to \Box\varphi)$ **&** $\Box(\psi \to \Box\psi)) \to ((\Box\varphi \equiv \Box\psi) \to \Box(\varphi \equiv \psi))$ *in* $v]$
proof (*rule CP*)
assume *1*: $[(\Box(\varphi \to \Box\varphi)$ **&** $\Box(\psi \to \Box\psi))$ *in* $v]$

```
    {
      assume [□φ ≡ □ψ in v]
      hence [(□φ & □ψ) ∨ ((¬(□φ)) & (¬(□ψ))) in v]
        using oth-class-taut-5-i[equiv-lr] by auto
      moreover {
        assume [□φ & □ψ in v]
        hence [□(φ ≡ ψ) in v]
          using KBasic-7[deduction] by auto
      }
      moreover {
        assume [(¬(□φ)) & (¬(□ψ)) in v]
        hence [□(¬φ) & □(¬ψ) in v]
            using 1 &E &I sc-eq-box-box-2[deduction, equiv-lr]
            by metis
        hence [□((¬φ) & (¬ψ)) in v]
          using KBasic-3[equiv-rl] by auto
        hence [□(φ ≡ ψ) in v]
          using KBasic-9[deduction] by auto
      }
      ultimately have [□(φ ≡ ψ) in v]
        using CP ∨E(1) by blast
    }
    thus [□φ ≡ □ψ → □(φ ≡ ψ) in v]
      using CP by auto
  qed

lemma derived-S5-rules-1-a[PLM]:
  assumes ⋀v. [χ in v] ⟹ [◇φ → ψ in v]
  shows [□χ in v] ⟹ [φ → □ψ in v]
  proof −
    have [□χ in v] ⟹ [□◇φ → □ψ in v]
      using assms RM-1-b by metis
    thus [□χ in v] ⟹ [φ → □ψ in v]
      using S5Basic-4 vdash-properties-10 CP by metis
  qed

lemma derived-S5-rules-1-b[PLM]:
  assumes ⋀v. [◇φ → ψ in v]
  shows [φ → □ψ in v]
  using derived-S5-rules-1-a all-self-eq-1 assms by blast

lemma derived-S5-rules-2-a[PLM]:
  assumes ⋀v. [χ in v] ⟹ [φ → □ψ in v]
  shows [□χ in v] ⟹ [◇φ → ψ in v]
  proof −
    have [□χ in v] ⟹ [◇φ → ◇□ψ in v]
      using RM-2-b assms by metis
    thus [□χ in v] ⟹ [◇φ → ψ in v]
      using B◇ vdash-properties-10 CP by metis
  qed

lemma derived-S5-rules-2-b[PLM]:
  assumes ⋀v. [φ → □ψ in v]
  shows [◇φ → ψ in v]
  using assms derived-S5-rules-2-a all-self-eq-1 by blast

lemma BFs-1[PLM]: [(∀ α. □(φ α)) → □(∀ α. φ α) in v]
  proof (rule derived-S5-rules-1-b)
    fix v
    {
      fix α
      have ⋀v.[(∀ α . □(φ α)) → □(φ α) in v]
        using cqt-orig-1 by metis
```

    hence $[\Diamond(\forall\,\alpha.\ \Box(\varphi\ \alpha)) \rightarrow \Diamond\Box(\varphi\ \alpha)\ \ in\ v]$
      using *RM-2* **by** *metis*
    moreover have $[\Diamond\Box(\varphi\ \alpha) \rightarrow (\varphi\ \alpha)\ \ in\ v]$
      using *B*$\Diamond$ **by** *auto*
    ultimately have $[\Diamond(\forall\,\alpha.\ \Box(\varphi\ \alpha)) \rightarrow (\varphi\ \alpha)\ \ in\ v]$
      using *ded-thm-cor-3* **by** *auto*
  }
  hence $[\forall\ \alpha\ .\ \Diamond(\forall\,\alpha.\ \Box(\varphi\ \alpha)) \rightarrow (\varphi\ \alpha)\ \ in\ v]$
    using $\forall\,I$ **by** *metis*
  thus $[\Diamond(\forall\,\alpha.\ \Box(\varphi\ \alpha)) \rightarrow (\forall\,\alpha.\ \varphi\ \alpha)\ \ in\ v]$
    using *cqt-orig-2*[*deduction*] **by** *auto*
qed
lemmas *BF* = *BFs-1*

lemma *BFs-2*[*PLM*]:
$[\Box(\forall\,\alpha.\ \varphi\ \alpha) \rightarrow (\forall\,\alpha.\ \Box(\varphi\ \alpha))\ \ in\ v]$
proof −
  {
    fix α
    {
      fix v
      have $[(\forall\,\alpha.\ \varphi\ \alpha) \rightarrow \varphi\ \alpha\ \ in\ v]$ **using** *cqt-orig-1* **by** *metis*
    }
    hence $[\Box(\forall\,\alpha\ .\ \varphi\ \alpha) \rightarrow \Box(\varphi\ \alpha)\ \ in\ v]$ **using** *RM-1* **by** *auto*
  }
  hence $[\forall\,\alpha\ .\ \Box(\forall\,\alpha\ .\ \varphi\ \alpha) \rightarrow \Box(\varphi\ \alpha)\ \ in\ v]$ **using** $\forall\,I$ **by** *metis*
  thus *?thesis* **using** *cqt-orig-2*[*deduction*] **by** *metis*
qed
lemmas *CBF* = *BFs-2*

lemma *BFs-3*[*PLM*]:
$[\Diamond(\exists\ \alpha.\ \varphi\ \alpha) \rightarrow (\exists\ \alpha\ .\ \Diamond(\varphi\ \alpha))\ \ in\ v]$
proof −
  have $[(\forall\,\alpha.\ \Box(\neg(\varphi\ \alpha))) \rightarrow \Box(\forall\,\alpha.\ \neg(\varphi\ \alpha))\ \ in\ v]$
    using *BF* **by** *metis*
  hence *1*: $[(\neg(\Box(\forall\,\alpha.\ \neg(\varphi\ \alpha)))) \rightarrow (\neg(\forall\,\alpha.\ \Box(\neg(\varphi\ \alpha))))\ \ in\ v]$
    using *contraposition-1* **by** *simp*
  have *2*: $[\Diamond(\neg(\forall\,\alpha.\ \neg(\varphi\ \alpha))) \rightarrow (\neg(\forall\,\alpha.\ \Box(\neg(\varphi\ \alpha))))\ \ in\ v]$
    apply (*PLM-subst-method* $\neg\Box(\forall\,\alpha\ .\ \neg(\varphi\ \alpha))\ \Diamond(\neg(\forall\,\alpha.\ \neg(\varphi\ \alpha)))$)
    using *KBasic2-2 1* **by** *simp+*
  have $[\Diamond(\neg(\forall\,\alpha.\ \neg(\varphi\ \alpha))) \rightarrow (\exists\ \alpha\ .\ \neg(\Box(\neg(\varphi\ \alpha))))\ \ in\ v]$
    apply (*PLM-subst-method* $\neg(\forall\,\alpha.\ \Box(\neg(\varphi\ \alpha)))\ \exists\ \alpha.\ \neg(\Box(\neg(\varphi\ \alpha)))$)
     using *cqt-further-2* **apply** *metis*
    using *2* **by** *metis*
  thus *?thesis*
    unfolding *exists-def diamond-def* **by** *auto*
qed
lemmas *BF*$\Diamond$ = *BFs-3*

lemma *BFs-4*[*PLM*]:
$[(\exists\ \alpha\ .\ \Diamond(\varphi\ \alpha)) \rightarrow \Diamond(\exists\ \alpha.\ \varphi\ \alpha)\ \ in\ v]$
proof −
  have *1*: $[\Box(\forall\,\alpha\ .\ \neg(\varphi\ \alpha)) \rightarrow (\forall\,\alpha.\ \Box(\neg(\varphi\ \alpha)))\ \ in\ v]$
    using *CBF* **by** *auto*
  have *2*: $[(\exists\ \alpha\ .\ (\neg(\Box(\neg(\varphi\ \alpha))))) \rightarrow (\neg(\Box(\forall\,\alpha.\ \neg(\varphi\ \alpha))))\ \ in\ v]$
    apply (*PLM-subst-method* $\neg(\forall\,\alpha.\ \Box(\neg(\varphi\ \alpha)))\ (\exists\ \alpha\ .\ (\neg(\Box(\neg(\varphi\ \alpha)))))$)
     using *cqt-further-2* **apply** *blast*
    using *1* **using** *contraposition-1* **by** *metis*
  have $[(\exists\ \alpha\ .\ (\neg(\Box(\neg(\varphi\ \alpha))))) \rightarrow \Diamond(\neg(\forall\ \alpha\ .\ \neg(\varphi\ \alpha)))\ \ in\ v]$
    apply (*PLM-subst-method* $\neg(\Box(\forall\,\alpha.\ \neg(\varphi\ \alpha)))\ \Diamond(\neg(\forall\,\alpha.\ \neg(\varphi\ \alpha)))$)
     using *KBasic2-2* **apply** *blast*
    using *2* **by** *assumption*
  thus *?thesis*

unfolding *diamond-def exists-def* **by** *auto*
qed
lemmas *CBF*$\Diamond$ = *BFs-4*

lemma *sign-S5-thm-1*[*PLM*]:
[($\exists$ α. $\Box$(φ α)) $\rightarrow$ $\Box$($\exists$ α. φ α) *in v*]
proof (*rule CP*)
assume [$\exists$ α . $\Box$(φ α) *in v*]
then obtain τ **where** [$\Box$(φ τ) *in v*]
by (*rule* $\exists$ *E*)
moreover {
fix *v*
assume [φ τ *in v*]
hence [$\exists$ α . φ α *in v*]
by (*rule* $\exists$ *I*)
}
ultimately show [$\Box$($\exists$ α . φ α) *in v*]
using *RN-2* **by** *blast*
qed
lemmas *Buridan* = *sign-S5-thm-1*

lemma *sign-S5-thm-2*[*PLM*]:
[$\Diamond$($\forall$ α . φ α) $\rightarrow$ ($\forall$ α . $\Diamond$(φ α)) *in v*]
proof −
{
fix α
{
fix *v*
have [($\forall$ α . φ α) $\rightarrow$ φ α *in v*]
using *cqt-orig-1* **by** *metis*
}
hence [$\Diamond$($\forall$ α . φ α) $\rightarrow$ $\Diamond$(φ α) *in v*]
using *RM-2* **by** *metis*
}
hence [$\forall$ α . $\Diamond$($\forall$ α . φ α) $\rightarrow$ $\Diamond$(φ α) *in v*]
using $\forall$ *I* **by** *metis*
thus *?thesis*
using *cqt-orig-2*[*deduction*] **by** *metis*
qed
lemmas *Buridan*$\Diamond$ = *sign-S5-thm-2*

lemma *sign-S5-thm-3*[*PLM*]:
[$\Diamond$($\exists$ α . φ α **&** ψ α) $\rightarrow$ $\Diamond$(($\exists$ α . φ α) **&** ($\exists$ α . ψ α)) *in v*]
by (*simp only*: *RM-2 cqt-further-5*)

lemma *sign-S5-thm-4*[*PLM*]:
[(($\Box$($\forall$ α. φ α $\rightarrow$ ψ α)) **&** ($\Box$($\forall$ α . ψ α $\rightarrow$ χ α))) $\rightarrow$ $\Box$($\forall$ α. φ α $\rightarrow$ χ α) *in v*]
proof (*rule CP*)
assume [$\Box$($\forall$ α. φ α $\rightarrow$ ψ α) **&** $\Box$($\forall$ α. ψ α $\rightarrow$ χ α) *in v*]
hence [$\Box$(($\forall$ α. φ α $\rightarrow$ ψ α) **&** ($\forall$ α. ψ α $\rightarrow$ χ α)) *in v*]
using *KBasic-3*[*equiv-rl*] **by** *blast*
moreover {
fix *v*
assume [(($\forall$ α. φ α $\rightarrow$ ψ α) **&** ($\forall$ α. ψ α $\rightarrow$ χ α)) *in v*]
hence [($\forall$ α . φ α $\rightarrow$ χ α) *in v*]
using *cqt-basic-9*[*deduction*] **by** *blast*
}
ultimately show [$\Box$($\forall$ α. φ α $\rightarrow$ χ α) *in v*]
using *RN-2* **by** *blast*
qed

lemma *sign-S5-thm-5*[*PLM*]:
[(($\Box$($\forall$ α. φ α $\equiv$ ψ α)) **&** ($\Box$($\forall$ α. ψ α $\equiv$ χ α))) $\rightarrow$ ($\Box$($\forall$ α. φ α $\equiv$ χ α)) *in v*]

proof (*rule CP*)
  assume [$\Box(\forall\,\alpha.\ \varphi\ \alpha \equiv \psi\ \alpha)$ **&** $\Box(\forall\,\alpha.\ \psi\ \alpha \equiv \chi\ \alpha)$ *in v*]
  hence [$\Box((\forall\,\alpha.\ \varphi\ \alpha \equiv \psi\ \alpha)$ **&** $(\forall\,\alpha.\ \psi\ \alpha \equiv \chi\ \alpha))$ *in v*]
    using *KBasic-3*[*equiv-rl*] **by** *blast*
  moreover {
    fix *v*
    assume [$((\forall\,\alpha.\ \varphi\ \alpha \equiv \psi\ \alpha)$ **&** $(\forall\,\alpha.\ \psi\ \alpha \equiv \chi\ \alpha))$ *in v*]
    hence [$(\forall\ \ \alpha\ .\ \varphi\ \alpha \equiv \chi\ \alpha)$ *in v*]
      using *cqt-basic-10*[*deduction*] **by** *blast*
  }
  ultimately show [$\Box(\forall\,\alpha.\ \varphi\ \alpha \equiv \chi\ \alpha)$ *in v*]
    using *RN-2* **by** *blast*
qed

lemma *id-nec2-1*[*PLM*]:
  [$\Diamond((\alpha::'a::id\text{-}eq) = \beta) \equiv (\alpha = \beta)$ *in v*]
  apply (*rule* $\equiv I$; *rule CP*)
   using *id-nec*[*equiv-lr*] *derived-S5-rules-2-b CP modus-ponens* **apply** *blast*
  using $T\Diamond$[*deduction*] **by** *auto*

lemma *id-nec2-2-Aux*:
  [$(\Diamond\varphi) \equiv \psi$ *in v*] $\Longrightarrow$ [$(\neg\psi) \equiv \Box(\neg\varphi)$ *in v*]
  proof −
    assume [$(\Diamond\varphi) \equiv \psi$ *in v*]
    moreover have $\bigwedge\varphi\ \psi$. [$(\neg\varphi) \equiv \psi$ *in v*] $\Longrightarrow$ [$(\neg\psi) \equiv \varphi$ *in v*]
      by *PLM-solver*
    ultimately show *?thesis*
      unfolding *diamond-def* **by** *blast*
  qed

lemma *id-nec2-2*[*PLM*]:
  [$((\alpha::'a::id\text{-}eq) \neq \beta) \equiv \Box(\alpha \neq \beta)$ *in v*]
  using *id-nec2-1*[*THEN id-nec2-2-Aux*] **by** *auto*

lemma *id-nec2-3*[*PLM*]:
  [$(\Diamond((\alpha::'a::id\text{-}eq) \neq \beta)) \equiv (\alpha \neq \beta)$ *in v*]
  using $T\Diamond$ $\equiv I$ *id-nec2-2*[*equiv-lr*]
    CP derived-S5-rules-2-b **by** *metis*

lemma *exists-desc-box-1*[*PLM*]:
  [$(\exists\ \ y\ .\ (y^P) = (\iota x.\ \varphi\ x)) \to (\exists\ \ y\ .\ \Box((y^P) = (\iota x.\ \varphi\ x)))$ *in v*]
  proof (*rule CP*)
    assume [$\exists\,y.\ (y^P) = (\iota x.\ \varphi\ x)$ *in v*]
    then obtain *y* **where** [$(y^P) = (\iota x.\ \varphi\ x)$ *in v*]
      by (*rule* $\exists\,E$)
    hence [$\Box(y^P = (\iota x.\ \varphi\ x))$ *in v*]
      using *l-identity*[*axiom-instance*, *deduction*, *deduction*]
        cqt-1[*axiom-instance*] *all-self-eq-2*[**where** $'a{=}\nu$]
        modus-ponens **unfolding** *identity-ν-def* **by** *fast*
    thus [$\exists\,y.\ \Box((y^P) = (\iota x.\ \varphi\ x))$ *in v*]
      by (*rule* $\exists\,I$)
  qed

lemma *exists-desc-box-2*[*PLM*]:
  [$(\exists\ \ y\ .\ (y^P) = (\iota x.\ \varphi\ x)) \to\ \Box(\exists\ \ y\ .((y^P) = (\iota x.\ \varphi\ x)))$ *in v*]
  using *exists-desc-box-1 Buridan ded-thm-cor-3* **by** *fast*

lemma *en-eq-1*[*PLM*]:
  [$\Diamond\{\!|x,F|\!\} \equiv \Box\{\!|x,F|\!\}$ *in v*]
  using *encoding*[*axiom-instance*] *RN*
    sc-eq-box-box-1 modus-ponens **by** *blast*
lemma *en-eq-2*[*PLM*]:
  [$\{\!|x,F|\!\} \equiv \Box\{\!|x,F|\!\}$ *in v*]

using *encoding*[*axiom-instance*] *qml-2*[*axiom-instance*] **by** (*rule* $\equiv I$)
lemma *en-eq-3*[*PLM*]:
[$\Diamond\{\!|x,F|\!\} \equiv \{\!|x,F|\!\}$ *in v*]
using *encoding*[*axiom-instance*] *derived-S5-rules-2-b* $\equiv I$ $T\Diamond$ **by** *auto*
lemma *en-eq-4*[*PLM*]:
[$(\{\!|x,F|\!\} \equiv \{\!|y,G|\!\}) \equiv (\Box\{\!|x,F|\!\} \equiv \Box\{\!|y,G|\!\})$ *in v*]
by (*metis CP en-eq-2* $\equiv I$ $\equiv E(1)$ $\equiv E(2)$)
lemma *en-eq-5*[*PLM*]:
[$\Box(\{\!|x,F|\!\} \equiv \{\!|y,G|\!\}) \equiv (\Box\{\!|x,F|\!\} \equiv \Box\{\!|y,G|\!\})$ *in v*]
using $\equiv I$ *KBasic-6 encoding*[*axiom-necessitation*, *axiom-instance*]
sc-eq-box-box-3[*deduction*] **&***I* **by** *simp*
lemma *en-eq-6*[*PLM*]:
[$(\{\!|x,F|\!\} \equiv \{\!|y,G|\!\}) \equiv \Box(\{\!|x,F|\!\} \equiv \{\!|y,G|\!\})$ *in v*]
using *en-eq-4 en-eq-5 oth-class-taut-4-a* $\equiv E(6)$ **by** *meson*
lemma *en-eq-7*[*PLM*]:
[$(\neg\{\!|x,F|\!\}) \equiv \Box(\neg\{\!|x,F|\!\})$ *in v*]
using *en-eq-3*[*THEN id-nec2-2-Aux*] **by** *blast*
lemma *en-eq-8*[*PLM*]:
[$\Diamond(\neg\{\!|x,F|\!\}) \equiv (\neg\{\!|x,F|\!\})$ *in v*]
unfolding *diamond-def* **apply** (*PLM-subst-method* $\{\!|x,F|\!\}$ $\neg\neg\{\!|x,F|\!\}$)
using *oth-class-taut-4-b* **apply** *simp*
apply (*PLM-subst-method* $\{\!|x,F|\!\}$ $\Box\{\!|x,F|\!\}$)
using *en-eq-2* **apply** *simp*
using *oth-class-taut-4-a* **by** *assumption*
lemma *en-eq-9*[*PLM*]:
[$\Diamond(\neg\{\!|x,F|\!\}) \equiv \Box(\neg\{\!|x,F|\!\})$ *in v*]
using *en-eq-8 en-eq-7* $\equiv E(5)$ **by** *blast*
lemma *en-eq-10*[*PLM*]:
[$\boldsymbol{\mathcal{A}}\{\!|x,F|\!\} \equiv \{\!|x,F|\!\}$ *in v*]
apply (*rule* $\equiv I$)
using *encoding*[*axiom-actualization*, *axiom-instance*,
THEN logic-actual-nec-2[*axiom-instance*, *equiv-lr*],
deduction, *THEN qml-act-2*[*axiom-instance*, *equiv-rl*],
THEN en-eq-2[*equiv-rl*]] *CP*
apply *simp*
using *encoding*[*axiom-instance*] *nec-imp-act ded-thm-cor-3* **by** *blast*

9.11 The Theory of Relations

lemma *beta-equiv-eq-1-1*[*PLM*]:
assumes *IsProperInX* φ
and *IsProperInX* ψ
and $\bigwedge x.[\varphi\ (x^P) \equiv \psi\ (x^P)$ *in v*]
shows [$(\!|\boldsymbol{\lambda}\ y.\ \varphi\ (y^P),\ x^P|\!) \equiv (\!|\boldsymbol{\lambda}\ y.\ \psi\ (y^P),\ x^P|\!)$ *in v*]
using *lambda-predicates-2-1*[*OF assms*(*1*), *axiom-instance*]
using *lambda-predicates-2-1*[*OF assms*(*2*), *axiom-instance*]
using *assms*(*3*) **by** (*meson* $\equiv E(6)$ *oth-class-taut-4-a*)

lemma *beta-equiv-eq-1-2*[*PLM*]:
assumes *IsProperInXY* φ
and *IsProperInXY* ψ
and $\bigwedge x\ y.[\varphi\ (x^P)\ (y^P) \equiv \psi\ (x^P)\ (y^P)$ *in v*]
shows [$(\!|\boldsymbol{\lambda}^2\ (\lambda\ x\ y.\ \varphi\ (x^P)\ (y^P)),\ x^P,\ y^P|\!)$
$\equiv (\!|\boldsymbol{\lambda}^2\ (\lambda\ x\ y.\ \psi\ (x^P)\ (y^P)),\ x^P,\ y^P|\!)$ *in v*]
using *lambda-predicates-2-2*[*OF assms*(*1*), *axiom-instance*]
using *lambda-predicates-2-2*[*OF assms*(*2*), *axiom-instance*]
using *assms*(*3*) **by** (*meson* $\equiv E(6)$ *oth-class-taut-4-a*)

lemma *beta-equiv-eq-1-3*[*PLM*]:
assumes *IsProperInXYZ* φ
and *IsProperInXYZ* ψ
and $\bigwedge x\ y\ z.[\varphi\ (x^P)\ (y^P)\ (z^P) \equiv \psi\ (x^P)\ (y^P)\ (z^P)$ *in v*]
shows [$(\!|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|\!)$

$\equiv (|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \psi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|)\ \mathit{in}\ v]$
using *lambda-predicates-2-3*[*OF assms(1), axiom-instance*]
using *lambda-predicates-2-3*[*OF assms(2), axiom-instance*]
using *assms(3)* **by** (*meson* $\equiv E(6)$ *oth-class-taut-4-a*)

lemma *beta-equiv-eq-2-1*[*PLM*]:
assumes *IsProperInX* φ
and *IsProperInX* ψ
shows $[(\Box(\forall\ x\ .\ \varphi\ (x^P) \equiv \psi\ (x^P))) \rightarrow$
$(\Box(\forall\ x\ .\ (|\boldsymbol{\lambda}\ y.\ \varphi\ (y^P),\ x^P|) \equiv (|\boldsymbol{\lambda}\ y.\ \psi\ (y^P),\ x^P|)))\ \mathit{in}\ v]$
apply (*rule qml-1*[*axiom-instance, deduction*])
apply (*rule RN*)
proof (*rule CP, rule* $\forall I$)
fix *v x*
assume $[\forall x.\ \varphi\ (x^P) \equiv \psi\ (x^P)\ \mathit{in}\ v]$
hence $\bigwedge x.[\varphi\ (x^P) \equiv \psi\ (x^P)\ \mathit{in}\ v]$
by *PLM-solver*
thus $[(|\boldsymbol{\lambda}\ y.\ \varphi\ (y^P),\ x^P|) \equiv (|\boldsymbol{\lambda}\ y.\ \psi\ (y^P),\ x^P|)\ \mathit{in}\ v]$
using *assms beta-equiv-eq-1-1* **by** *auto*
qed

lemma *beta-equiv-eq-2-2*[*PLM*]:
assumes *IsProperInXY* φ
and *IsProperInXY* ψ
shows $[(\Box(\forall\ x\ y\ .\ \varphi\ (x^P)\ (y^P) \equiv \psi\ (x^P)\ (y^P))) \rightarrow$
$(\Box(\forall\ x\ y\ .\ (|\boldsymbol{\lambda}^2\ (\lambda\ x\ y.\ \varphi\ (x^P)\ (y^P)),\ x^P,\ y^P|)$
$\equiv (|\boldsymbol{\lambda}^2\ (\lambda\ x\ y.\ \psi\ (x^P)\ (y^P)),\ x^P,\ y^P|)))\ \mathit{in}\ v]$
apply (*rule qml-1*[*axiom-instance, deduction*])
apply (*rule RN*)
proof (*rule CP, rule* $\forall I$, *rule* $\forall I$)
fix *v x y*
assume $[\forall x\ y.\ \varphi\ (x^P)\ (y^P) \equiv \psi\ (x^P)\ (y^P)\ \mathit{in}\ v]$
hence $(\bigwedge x\ y.[\varphi\ (x^P)\ (y^P) \equiv \psi\ (x^P)\ (y^P)\ \mathit{in}\ v])$
by (*meson* $\forall E$)
thus $[(|\boldsymbol{\lambda}^2\ (\lambda\ x\ y.\ \varphi\ (x^P)\ (y^P)),\ x^P,\ y^P|)$
$\equiv (|\boldsymbol{\lambda}^2\ (\lambda\ x\ y.\ \psi\ (x^P)\ (y^P)),\ x^P,\ y^P|)\ \mathit{in}\ v]$
using *assms beta-equiv-eq-1-2* **by** *auto*
qed

lemma *beta-equiv-eq-2-3*[*PLM*]:
assumes *IsProperInXYZ* φ
and *IsProperInXYZ* ψ
shows $[(\Box(\forall\ x\ y\ z\ .\ \varphi\ (x^P)\ (y^P)\ (z^P) \equiv \psi\ (x^P)\ (y^P)\ (z^P))) \rightarrow$
$(\Box(\forall\ x\ y\ z\ .\ (|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|)$
$\equiv (|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \psi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|)))\ \mathit{in}\ v]$
apply (*rule qml-1*[*axiom-instance, deduction*])
apply (*rule RN*)
proof (*rule CP, rule* $\forall I$, *rule* $\forall I$, *rule* $\forall I$)
fix *v x y z*
assume $[\forall x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P) \equiv \psi\ (x^P)\ (y^P)\ (z^P)\ \mathit{in}\ v]$
hence $(\bigwedge x\ y\ z.[\varphi\ (x^P)\ (y^P)\ (z^P) \equiv \psi\ (x^P)\ (y^P)\ (z^P)\ \mathit{in}\ v])$
by (*meson* $\forall E$)
thus $[(|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|)$
$\equiv (|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \psi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|)\ \mathit{in}\ v]$
using *assms beta-equiv-eq-1-3* **by** *auto*
qed

lemma *beta-C-meta-1*[*PLM*]:
assumes *IsProperInX* φ
shows $[(|\boldsymbol{\lambda}\ y.\ \varphi\ (y^P),\ x^P|) \equiv \varphi\ (x^P)\ \mathit{in}\ v]$
using *lambda-predicates-2-1*[*OF assms, axiom-instance*] **by** *auto*

lemma *beta-C-meta-2*[*PLM*]:

assumes *IsProperInXY* φ
shows $[(\!|\boldsymbol{\lambda}^2\ (\lambda\ x\ y.\ \varphi\ (x^P)\ (y^P)),\ x^P,\ y^P|\!) \equiv \varphi\ (x^P)\ (y^P)\ in\ v]$
using *lambda-predicates-2-2*[*OF assms*, *axiom-instance*] **by** *auto*

lemma *beta-C-meta-3*[*PLM*]:
assumes *IsProperInXYZ* φ
shows $[(\!|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|\!) \equiv \varphi\ (x^P)\ (y^P)\ (z^P)\ in\ v]$
using *lambda-predicates-2-3*[*OF assms*, *axiom-instance*] **by** *auto*

lemma *relations-1*[*PLM*]:
assumes *IsProperInX* φ
shows $[\exists\ F.\ \Box(\forall\ x.\ (\!|F,x^P|\!) \equiv \varphi\ (x^P))\ in\ v]$
using *assms* **apply** − **by** *PLM-solver*

lemma *relations-2*[*PLM*]:
assumes *IsProperInXY* φ
shows $[\exists\ F.\ \Box(\forall\ x\ y.\ (\!|F,x^P,y^P|\!) \equiv \varphi\ (x^P)\ (y^P))\ in\ v]$
using *assms* **apply** − **by** *PLM-solver*

lemma *relations-3*[*PLM*]:
assumes *IsProperInXYZ* φ
shows $[\exists\ F.\ \Box(\forall\ x\ y\ z.\ (\!|F,x^P,y^P,z^P|\!) \equiv \varphi\ (x^P)\ (y^P)\ (z^P))\ in\ v]$
using *assms* **apply** − **by** *PLM-solver*

lemma *prop-equiv*[*PLM*]:
shows $[(\forall\ x\ .\ (\{\!|x^P,F|\!\} \equiv \{\!|x^P,G|\!\})) \rightarrow F = G\ in\ v]$
proof (*rule CP*)
assume *1*: $[\forall x.\ \{\!|x^P,F|\!\} \equiv \{\!|x^P,G|\!\}\ in\ v]$
{
fix *x*
have $[\{\!|x^P,F|\!\} \equiv \{\!|x^P,G|\!\}\ in\ v]$
using *1* **by** (*rule* $\forall E$)
hence $[\Box(\{\!|x^P,F|\!\} \equiv \{\!|x^P,G|\!\})\ in\ v]$
using *PLM.en-eq-6* $\equiv E(1)$ **by** *blast*
}
hence $[\forall x.\ \Box(\{\!|x^P,F|\!\} \equiv \{\!|x^P,G|\!\})\ in\ v]$
by (*rule* $\forall I$)
thus $[F = G\ in\ v]$
unfolding *identity-defs*
by (*rule BF*[*deduction*])
qed

lemma *propositions-lemma-1*[*PLM*]:
$[\boldsymbol{\lambda}^0\ \varphi = \varphi\ in\ v]$
using *lambda-predicates-3-0*[*axiom-instance*] **.**

lemma *propositions-lemma-2*[*PLM*]:
$[\boldsymbol{\lambda}^0\ \varphi \equiv \varphi\ in\ v]$
using *lambda-predicates-3-0*[*axiom-instance*, *THEN id-eq-prop-prop-8-b*[*deduction*]]
apply (*rule l-identity*[*axiom-instance*, *deduction*, *deduction*])
by *PLM-solver*

lemma *propositions-lemma-4*[*PLM*]:
assumes $\bigwedge x.[\boldsymbol{\mathcal{A}}(\varphi\ x \equiv \psi\ x)\ in\ v]$
shows $[(\chi{::}\kappa{\Rightarrow}\text{o})\ (\iota x.\ \varphi\ x) = \chi\ (\iota x.\ \psi\ x)\ in\ v]$
proof −
have $[\boldsymbol{\lambda}^0\ (\chi\ (\iota x.\ \varphi\ x)) = \boldsymbol{\lambda}^0\ (\chi\ (\iota x.\ \psi\ x))\ in\ v]$
using *assms lambda-predicates-4-0*[*axiom-instance*]
by *blast*
hence $[(\chi\ (\iota x.\ \varphi\ x)) = \boldsymbol{\lambda}^0\ (\chi\ (\iota x.\ \psi\ x))\ in\ v]$
using *propositions-lemma-1*[*THEN id-eq-prop-prop-8-b*[*deduction*]]
id-eq-prop-prop-9-b[*deduction*] **&***I*
by *blast*

thus *?thesis*
using *propositions-lemma-1 id-eq-prop-prop-9-b*[*deduction*] &I
by *blast*
qed

lemma *propositions*[*PLM*]:
$[\exists\ p\ .\ \Box(p \equiv p')\ in\ v]$
by *PLM-solver*

lemma *pos-not-equiv-then-not-eq*[*PLM*]:
$[\Diamond(\neg(\forall\, x.\ (\!|F,x^P|\!) \equiv (\!|G,x^P|\!))) \to F \neq G\ in\ v]$
unfolding *diamond-def*
proof (*subst contraposition-1*[*symmetric*], *rule CP*)
assume $[F = G\ in\ v]$
thus $[\Box(\neg(\neg(\forall\, x.\ (\!|F,x^P|\!) \equiv (\!|G,x^P|\!))))\ in\ v]$
apply (*rule l-identity*[*axiom-instance*, *deduction*, *deduction*])
by *PLM-solver*
qed

lemma *thm-relation-negation-1-1*[*PLM*]:
$[(\!|F^-,\ x^P|\!) \equiv \neg(\!|F,\ x^P|\!)\ in\ v]$
unfolding *propnot-defs*
apply (*rule lambda-predicates-2-1*[*axiom-instance*])
by *show-proper*

lemma *thm-relation-negation-1-2*[*PLM*]:
$[(\!|F^-,\ x^P,\ y^P|\!) \equiv \neg(\!|F,\ x^P,\ y^P|\!)\ in\ v]$
unfolding *propnot-defs*
apply (*rule lambda-predicates-2-2*[*axiom-instance*])
by *show-proper*

lemma *thm-relation-negation-1-3*[*PLM*]:
$[(\!|F^-,\ x^P,\ y^P,\ z^P|\!) \equiv \neg(\!|F,\ x^P,\ y^P,\ z^P|\!)\ in\ v]$
unfolding *propnot-defs*
apply (*rule lambda-predicates-2-3*[*axiom-instance*])
by *show-proper*

lemma *thm-relation-negation-2-1*[*PLM*]:
$[(\neg(\!|F^-,\ x^P|\!)) \equiv (\!|F,\ x^P|\!)\ in\ v]$
using *thm-relation-negation-1-1*[*THEN oth-class-taut-5-d*[*equiv-lr*]]
apply $-$ **by** *PLM-solver*

lemma *thm-relation-negation-2-2*[*PLM*]:
$[(\neg(\!|F^-,\ x^P,\ y^P|\!)) \equiv (\!|F,\ x^P,\ y^P|\!)\ in\ v]$
using *thm-relation-negation-1-2*[*THEN oth-class-taut-5-d*[*equiv-lr*]]
apply $-$ **by** *PLM-solver*

lemma *thm-relation-negation-2-3*[*PLM*]:
$[(\neg(\!|F^-,\ x^P,\ y^P,\ z^P|\!)) \equiv (\!|F,\ x^P,\ y^P,\ z^P|\!)\ in\ v]$
using *thm-relation-negation-1-3*[*THEN oth-class-taut-5-d*[*equiv-lr*]]
apply $-$ **by** *PLM-solver*

lemma *thm-relation-negation-3*[*PLM*]:
$[(p)^- \equiv \neg p\ in\ v]$
unfolding *propnot-defs*
using *propositions-lemma-2* **by** *simp*

lemma *thm-relation-negation-4*[*PLM*]:
$[(\neg((p{::}\mathrm{o})^-)) \equiv p\ in\ v]$
using *thm-relation-negation-3*[*THEN oth-class-taut-5-d*[*equiv-lr*]]
apply $-$ **by** *PLM-solver*

lemma *thm-relation-negation-5-1*[*PLM*]:

$[(F::\Pi_1) \neq (F^-)$ *in* $v]$
using *id-eq-prop-prop-2*[*deduction*]
l-identity[**where** $\varphi=\lambda\ G\ .\ (|G,x^P|) \equiv (|F^-,x^P|)$, *axiom-instance*, *deduction*, *deduction*]
oth-class-taut-4-a thm-relation-negation-1-1 $\equiv E(5)$
oth-class-taut-1-b modus-tollens-1 CP
by *meson*

lemma *thm-relation-negation-5-2*[*PLM*]:
$[(F::\Pi_2) \neq (F^-)$ *in* $v]$
using *id-eq-prop-prop-5-a*[*deduction*]
l-identity[**where** $\varphi=\lambda\ G\ .\ (|G,x^P,y^P|) \equiv (|F^-,x^P,y^P|)$, *axiom-instance*, *deduction*, *deduction*]
oth-class-taut-4-a thm-relation-negation-1-2 $\equiv E(5)$
oth-class-taut-1-b modus-tollens-1 CP
by *meson*

lemma *thm-relation-negation-5-3*[*PLM*]:
$[(F::\Pi_3) \neq (F^-)$ *in* $v]$
using *id-eq-prop-prop-5-b*[*deduction*]
l-identity[**where** $\varphi=\lambda\ G\ .\ (|G,x^P,y^P,z^P|) \equiv (|F^-,x^P,y^P,z^P|)$, *axiom-instance*, *deduction*, *deduction*]
oth-class-taut-4-a thm-relation-negation-1-3 $\equiv E(5)$
oth-class-taut-1-b modus-tollens-1 CP
by *meson*

lemma *thm-relation-negation-6*[*PLM*]:
$[(p::\mathrm{o}) \neq (p^-)$ *in* $v]$
using *id-eq-prop-prop-8-b*[*deduction*]
l-identity[**where** $\varphi=\lambda\ G\ .\ G \equiv (p^-)$, *axiom-instance*, *deduction*, *deduction*]
oth-class-taut-4-a thm-relation-negation-3 $\equiv E(5)$
oth-class-taut-1-b modus-tollens-1 CP
by *meson*

lemma *thm-relation-negation-7*[*PLM*]:
$[((p::\mathrm{o})^-) = \neg p$ *in* $v]$
unfolding *propnot-defs* **using** *propositions-lemma-1* **by** *simp*

lemma *thm-relation-negation-8*[*PLM*]:
$[(p::\mathrm{o}) \neq \neg p$ *in* $v]$
unfolding *propnot-defs*
using *id-eq-prop-prop-8-b*[*deduction*]
l-identity[**where** $\varphi=\lambda\ G\ .\ G \equiv \neg(p)$, *axiom-instance*, *deduction*, *deduction*]
oth-class-taut-4-a oth-class-taut-1-b
modus-tollens-1 CP
by *meson*

lemma *thm-relation-negation-9*[*PLM*]:
$[((p::\mathrm{o}) = q) \rightarrow ((\neg p) = (\neg q))$ *in* $v]$
using *l-identity*[**where** $\alpha=p$ **and** $\beta=q$ **and** $\varphi=\lambda\ x\ .\ (\neg p) = (\neg x)$, *axiom-instance*, *deduction*]
id-eq-prop-prop-7-b **using** *CP modus-ponens* **by** *blast*

lemma *thm-relation-negation-10*[*PLM*]:
$[((p::\mathrm{o}) = q) \rightarrow ((p^-) = (q^-))$ *in* $v]$
using *l-identity*[**where** $\alpha=p$ **and** $\beta=q$ **and** $\varphi=\lambda\ x\ .\ (p^-) = (x^-)$, *axiom-instance*, *deduction*]
id-eq-prop-prop-7-b **using** *CP modus-ponens* **by** *blast*

lemma *thm-cont-prop-1*[*PLM*]:
$[$*NonContingent* $(F::\Pi_1) \equiv$ *NonContingent* (F^-) *in* $v]$

proof ($rule \equiv I$; $rule$ CP)
assume [*NonContingent F in v*]
hence $[\Box(\forall x.\llparenthesis F,x^P\rrparenthesis) \lor \Box(\forall x.\neg\llparenthesis F,x^P\rrparenthesis)$ *in v*]
unfolding *NonContingent-def Necessary-defs Impossible-defs* **.**
hence $[\Box(\forall x.\ \neg\llparenthesis F^-,x^P\rrparenthesis) \lor \Box(\forall x.\ \neg\llparenthesis F,x^P\rrparenthesis)$ *in v*]
apply $-$
apply (*PLM-subst-method* $\lambda\ x\ .\ \llparenthesis F,x^P\rrparenthesis\ \lambda\ x\ .\ \neg\llparenthesis F^-,x^P\rrparenthesis$)
using *thm-relation-negation-2-1*[*equiv-sym*] **by** *auto*
hence $[\Box(\forall x.\ \neg\llparenthesis F^-,x^P\rrparenthesis) \lor \Box(\forall x.\ \llparenthesis F^-,x^P\rrparenthesis)$ *in v*]
apply $-$
apply (*PLM-subst-goal-method*
$\lambda\ \varphi\ .\ \Box(\forall x.\ \neg\llparenthesis F^-,x^P\rrparenthesis) \lor \Box(\forall x.\ \varphi\ x)\ \lambda\ x\ .\ \neg\llparenthesis F,x^P\rrparenthesis$)
using *thm-relation-negation-1-1*[*equiv-sym*] **by** *auto*
hence $[\Box(\forall x.\ \llparenthesis F^-,x^P\rrparenthesis) \lor \Box(\forall x.\ \neg\llparenthesis F^-,x^P\rrparenthesis)$ *in v*]
by (*rule oth-class-taut-3-e*[*equiv-lr*])
thus [*NonContingent* (F^-) *in v*]
unfolding *NonContingent-def Necessary-defs Impossible-defs* **.**
next
assume [*NonContingent* (F^-) *in v*]
hence $[\Box(\forall x.\ \neg\llparenthesis F^-,x^P\rrparenthesis) \lor \Box(\forall x.\ \llparenthesis F^-,x^P\rrparenthesis)$ *in v*]
unfolding *NonContingent-def Necessary-defs Impossible-defs*
by (*rule oth-class-taut-3-e*[*equiv-lr*])
hence $[\Box(\forall x.\llparenthesis F,x^P\rrparenthesis) \lor \Box(\forall x.\llparenthesis F^-,x^P\rrparenthesis)$ *in v*]
apply $-$
apply (*PLM-subst-method* $\lambda\ x\ .\ \neg\llparenthesis F^-,x^P\rrparenthesis\ \lambda\ x\ .\ \llparenthesis F,x^P\rrparenthesis$)
using *thm-relation-negation-2-1* **by** *auto*
hence $[\Box(\forall x.\ \llparenthesis F,x^P\rrparenthesis) \lor \Box(\forall x.\ \neg\llparenthesis F,x^P\rrparenthesis)$ *in v*]
apply $-$
apply (*PLM-subst-method* $\lambda\ x\ .\ \llparenthesis F^-,x^P\rrparenthesis\ \lambda\ x\ .\ \neg\llparenthesis F,x^P\rrparenthesis$)
using *thm-relation-negation-1-1* **by** *auto*
thus [*NonContingent F in v*]
unfolding *NonContingent-def Necessary-defs Impossible-defs* **.**
qed

lemma *thm-cont-prop-2*[*PLM*]:
[*Contingent* $F \equiv \Diamond(\exists\ x\ .\ \llparenthesis F,x^P\rrparenthesis)$ **&** $\Diamond(\exists\ x\ .\ \neg\llparenthesis F,x^P\rrparenthesis)$ *in v*]
proof ($rule \equiv I$; $rule$ CP)
assume [*Contingent F in v*]
hence $[\neg(\Box(\forall x.\llparenthesis F,x^P\rrparenthesis) \lor \Box(\forall x.\neg\llparenthesis F,x^P\rrparenthesis))$ *in v*]
unfolding *Contingent-def Necessary-defs Impossible-defs* **.**
hence $[(\neg\Box(\forall x.\llparenthesis F,x^P\rrparenthesis))$ **&** $(\neg\Box(\forall x.\neg\llparenthesis F,x^P\rrparenthesis))$ *in v*]
by (*rule oth-class-taut-6-d*[*equiv-lr*])
hence $[(\Diamond\neg(\forall x.\neg\llparenthesis F,x^P\rrparenthesis))$ **&** $(\Diamond\neg(\forall x.\llparenthesis F,x^P\rrparenthesis))$ *in v*]
using *KBasic2-2*[*equiv-lr*] **&**I **&**E **by** *meson*
thus $[(\Diamond(\exists\ x.\llparenthesis F,x^P\rrparenthesis))$ **&** $(\Diamond(\exists x.\ \neg\llparenthesis F,x^P\rrparenthesis))$ *in v*]
unfolding *exists-def* **apply** $-$
apply (*PLM-subst-method* $\lambda\ x\ .\ \llparenthesis F,x^P\rrparenthesis\ \lambda\ x\ .\ \neg\neg\llparenthesis F,x^P\rrparenthesis$)
using *oth-class-taut-4-b* **by** *auto*
next
assume $[(\Diamond(\exists\ x.\llparenthesis F,x^P\rrparenthesis))$ **&** $(\Diamond(\exists x.\ \neg\llparenthesis F,x^P\rrparenthesis))$ *in v*]
hence $[(\Diamond\neg(\forall x.\neg\llparenthesis F,x^P\rrparenthesis))$ **&** $(\Diamond\neg(\forall x.\llparenthesis F,x^P\rrparenthesis))$ *in v*]
unfolding *exists-def* **apply** $-$
apply (*PLM-subst-goal-method*
$\lambda\ \varphi\ .\ (\Diamond\neg(\forall x.\neg\llparenthesis F,x^P\rrparenthesis))$ **&** $(\Diamond\neg(\forall x.\ \varphi\ x))\ \lambda\ x\ .\ \neg\neg\llparenthesis F,x^P\rrparenthesis$)
using *oth-class-taut-4-b*[*equiv-sym*] **by** *auto*
hence $[(\neg\Box(\forall x.\llparenthesis F,x^P\rrparenthesis))$ **&** $(\neg\Box(\forall x.\neg\llparenthesis F,x^P\rrparenthesis))$ *in v*]
using *KBasic2-2*[*equiv-rl*] **&**I **&**E **by** *meson*
hence $[\neg(\Box(\forall x.\llparenthesis F,x^P\rrparenthesis) \lor \Box(\forall x.\neg\llparenthesis F,x^P\rrparenthesis))$ *in v*]
by (*rule oth-class-taut-6-d*[*equiv-rl*])
thus [*Contingent F in v*]
unfolding *Contingent-def Necessary-defs Impossible-defs* **.**
qed

lemma *thm-cont-prop-3*[*PLM*]:
[*Contingent* $(F::\Pi_1) \equiv$ *Contingent* (F^-) *in v*]
using *thm-cont-prop-1*
unfolding *NonContingent-def Contingent-def*
by (*rule oth-class-taut-5-d*[*equiv-lr*])

lemma *lem-cont-e*[*PLM*]:
$[\Diamond(\exists\ x\ .\ (\!|F,x^P|\!)\ \&\ (\Diamond(\neg(\!|F,x^P|\!)))) \equiv \Diamond(\exists\ x\ .\ ((\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!)))\mathit{in}\ v]$
proof −
have $[\Diamond(\exists\ x\ .\ (\!|F,x^P|\!)\ \&\ (\Diamond(\neg(\!|F,x^P|\!))))\ \mathit{in}\ v]$
$= [(\exists\ x\ .\ \Diamond((\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!))))\ \mathit{in}\ v]$
using *BF*$\Diamond$[*deduction*] *CBF*$\Diamond$[*deduction*] **by** *fast*
also have ... $= [\exists\ x\ .\ (\Diamond(\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!)))\ \mathit{in}\ v]$
apply (*PLM-subst-method*
$\lambda\ x\ .\ \Diamond((\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!)))$
$\lambda\ x\ .\ \Diamond(\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!)))$
using *S5Basic-12* **by** *auto*
also have ... $= [\exists\ x\ .\ \Diamond(\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!)\ \mathit{in}\ v]$
apply (*PLM-subst-method*
$\lambda\ x\ .\ \Diamond(\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!))$
$\lambda\ x\ .\ \Diamond(\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!))$
using *oth-class-taut-3-b* **by** *auto*
also have ... $= [\exists\ x\ .\ \Diamond((\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!))\ \mathit{in}\ v]$
apply (*PLM-subst-method*
$\lambda\ x\ .\ \Diamond(\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!)$
$\lambda\ x\ .\ \Diamond((\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!)))$
using *S5Basic-12*[*equiv-sym*] **by** *auto*
also have ... $= [\Diamond\ (\exists\ x\ .\ ((\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!)))\ \mathit{in}\ v]$
using *CBF*$\Diamond$[*deduction*] *BF*$\Diamond$[*deduction*] **by** *fast*
finally show *?thesis* **using** $\equiv I$ *CP* **by** *blast*
qed

lemma *lem-cont-e-2*[*PLM*]:
$[\Diamond(\exists\ x\ .\ (\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!))) \equiv \Diamond(\exists\ x\ .\ (\!|F^-,x^P|\!)\ \&\ \Diamond(\neg(\!|F^-,x^P|\!)))\ \mathit{in}\ v]$
apply (*PLM-subst-method* $\lambda\ x\ .\ (\!|F,x^P|\!)\ \lambda\ x\ .\ \neg(\!|F^-,x^P|\!))$
using *thm-relation-negation-2-1*[*equiv-sym*] **apply** *simp*
apply (*PLM-subst-method* $\lambda\ x\ .\ \neg(\!|F,x^P|\!)\ \lambda\ x\ .\ (\!|F^-,x^P|\!))$
using *thm-relation-negation-1-1*[*equiv-sym*] **apply** *simp*
using *lem-cont-e* **by** *simp*

lemma *thm-cont-e-1*[*PLM*]:
$[\Diamond(\exists\ x\ .\ ((\neg(\!|E!,x^P|\!))\ \&\ (\Diamond(\!|E!,x^P|\!))))\ \mathit{in}\ v]$
using *lem-cont-e*[**where** $F=E!$, *equiv-lr*] *qml-4*[*axiom-instance*,*conj1*]
by *blast*

lemma *thm-cont-e-2*[*PLM*]:
[*Contingent* $(E!)$ *in v*]
using *thm-cont-prop-2*[*equiv-rl*] $\&I$ *qml-4*[*axiom-instance*, *conj1*]
KBasic2-8[*deduction*, *OF sign-S5-thm-3*[*deduction*], *conj1*]
KBasic2-8[*deduction*, *OF sign-S5-thm-3*[*deduction*, *OF thm-cont-e-1*], *conj1*]
by *fast*

lemma *thm-cont-e-3*[*PLM*]:
[*Contingent* $(E!^-)$ *in v*]
using *thm-cont-e-2 thm-cont-prop-3*[*equiv-lr*] **by** *blast*

lemma *thm-cont-e-4*[*PLM*]:
$[\exists\ (F::\Pi_1)\ G\ .\ (F \neq G$ $\&$ *Contingent* F $\&$ *Contingent* $G)$ *in v*]
apply (*rule-tac* $\alpha=E!$ **in** $\exists I$, *rule-tac* $\alpha=E!^-$ **in** $\exists I$)
using *thm-cont-e-2 thm-cont-e-3 thm-relation-negation-5-1* $\&I$ **by** *auto*

context
begin

qualified definition L **where** $L \equiv (\boldsymbol{\lambda}\ x\ .\ (|E!,\ x^P|) \rightarrow (|E!,\ x^P|))$

lemma *thm-noncont-e-e-1*[*PLM*]:
[*Necessary L in v*]
unfolding *Necessary-defs L-def* **apply** (*rule RN*, *rule* ∀ *I*)
apply (*rule lambda-predicates-2-1*[*axiom-instance*, *equiv-rl*])
apply *show-proper*
using *if-p-then-p* **.**

lemma *thm-noncont-e-e-2*[*PLM*]:
[*Impossible* (L^-) *in v*]
unfolding *Impossible-defs L-def* **apply** (*rule RN*, *rule* ∀ *I*)
apply (*rule thm-relation-negation-2-1*[*equiv-rl*])
apply (*rule lambda-predicates-2-1*[*axiom-instance*, *equiv-rl*])
apply *show-proper*
using *if-p-then-p* **.**

lemma *thm-noncont-e-e-3*[*PLM*]:
[*NonContingent* (L) *in v*]
unfolding *NonContingent-def* **using** *thm-noncont-e-e-1*
by (*rule* ∨*I*(*1*))

lemma *thm-noncont-e-e-4*[*PLM*]:
[*NonContingent* (L^-) *in v*]
unfolding *NonContingent-def* **using** *thm-noncont-e-e-2*
by (*rule* ∨*I*(*2*))

lemma *thm-noncont-e-e-5*[*PLM*]:
[$\exists$ (F::Π_1) G . $F \neq G$ **&** *NonContingent F* **&** *NonContingent G in v*]
apply (*rule-tac* α=L **in** ∃ *I*, *rule-tac* α=L^- **in** ∃ *I*)
using ∃ *I thm-relation-negation-5-1 thm-noncont-e-e-3*
thm-noncont-e-e-4 **&***I*
by *simp*

lemma *four-distinct-1*[*PLM*]:
[*NonContingent* (F::Π_1) → ¬(∃ G . (*Contingent G* **&** $G = F$)) *in v*]
proof (*rule CP*)
assume [*NonContingent F in v*]
hence [¬(*Contingent F*) *in v*]
unfolding *NonContingent-def Contingent-def*
apply − **by** *PLM-solver*
moreover {
assume [∃ G . *Contingent G* **&** $G = F$ *in v*]
then obtain P **where** [*Contingent P* **&** $P = F$ *in v*]
by (*rule* ∃ *E*)
hence [*Contingent F in v*]
using &*E l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *blast*
}
ultimately show [¬(∃ *G*. *Contingent G* **&** $G = F$) *in v*]
using *modus-tollens-1 CP* **by** *blast*
qed

lemma *four-distinct-2*[*PLM*]:
[*Contingent* (F::Π_1) → ¬(∃ G . (*NonContingent G* **&** $G = F$)) *in v*]
proof (*rule CP*)
assume [*Contingent F in v*]
hence [¬(*NonContingent F*) *in v*]
unfolding *NonContingent-def Contingent-def*
apply − **by** *PLM-solver*
moreover {
assume [∃ G . *NonContingent G* **&** $G = F$ *in v*]

```
    then obtain P where [NonContingent P & P = F in v]
     by (rule ∃E)
    hence [NonContingent F in v]
      using &E l-identity[axiom-instance, deduction, deduction]
      by blast
  }
  ultimately show [¬(∃G. NonContingent G & G = F) in v]
    using modus-tollens-1 CP by blast
qed

lemma four-distinct-3[PLM]:
  [L ≠ (L⁻) & L ≠ E! & L ≠ (E!⁻) & (L⁻) ≠ E!
    & (L⁻) ≠ (E!⁻) & E! ≠ (E!⁻) in v]
  proof (rule &I)+
    show [L ≠ (L⁻) in v]
    by (rule thm-relation-negation-5-1)
  next
    {
      assume [L = E! in v]
      hence [NonContingent L & L = E! in v]
        using thm-noncont-e-e-3 &I by auto
      hence [∃ G . NonContingent G & G = E! in v]
        using thm-noncont-e-e-3 &I ∃I by fast
    }
    thus [L ≠ E! in v]
      using four-distinct-2[deduction, OF thm-cont-e-2]
            modus-tollens-1 CP
      by blast
  next
    {
      assume [L = (E!⁻) in v]
      hence [NonContingent L & L = (E!⁻) in v]
        using thm-noncont-e-e-3 &I by auto
      hence [∃ G . NonContingent G & G = (E!⁻) in v]
        using thm-noncont-e-e-3 &I ∃I by fast
    }
    thus [L ≠ (E!⁻) in v]
      using four-distinct-2[deduction, OF thm-cont-e-3]
            modus-tollens-1 CP
      by blast
  next
    {
      assume [(L⁻) = E! in v]
      hence [NonContingent (L⁻) & (L⁻) = E! in v]
        using thm-noncont-e-e-4 &I by auto
      hence [∃ G . NonContingent G & G = E! in v]
        using thm-noncont-e-e-3 &I ∃I by fast
    }
    thus [(L⁻) ≠ E! in v]
      using four-distinct-2[deduction, OF thm-cont-e-2]
            modus-tollens-1 CP
      by blast
  next
    {
      assume [(L⁻) = (E!⁻) in v]
      hence [NonContingent (L⁻) & (L⁻) = (E!⁻) in v]
        using thm-noncont-e-e-4 &I by auto
      hence [∃ G . NonContingent G & G = (E!⁻) in v]
        using thm-noncont-e-e-3 &I ∃I by fast
    }
    thus [(L⁻) ≠ (E!⁻) in v]
      using four-distinct-2[deduction, OF thm-cont-e-3]
            modus-tollens-1 CP
```

```
      by blast
    next
      show [E! ≠ (E!⁻) in v]
        by (rule thm-relation-negation-5-1)
    qed
end

lemma thm-cont-propos-1[PLM]:
  [NonContingent (p::o) ≡ NonContingent (p⁻) in v]
  proof (rule ≡I; rule CP)
    assume [NonContingent p in v]
    hence [□p ∨ □¬p in v]
      unfolding NonContingent-def Necessary-defs Impossible-defs .
    hence [□(¬(p⁻)) ∨ □(¬p) in v]
      apply -
      apply (PLM-subst-method p ¬(p⁻))
      using thm-relation-negation-4[equiv-sym] by auto
    hence [□(¬(p⁻)) ∨ □(p⁻) in v]
      apply -
      apply (PLM-subst-goal-method λφ . □(¬(p⁻)) ∨ □(φ) ¬p)
      using thm-relation-negation-3[equiv-sym] by auto
    hence [□(p⁻) ∨ □(¬(p⁻)) in v]
      by (rule oth-class-taut-3-e[equiv-lr])
    thus [NonContingent (p⁻) in v]
      unfolding NonContingent-def Necessary-defs Impossible-defs .
  next
    assume [NonContingent (p⁻) in v]
    hence [□(¬(p⁻)) ∨ □(p⁻) in v]
      unfolding NonContingent-def Necessary-defs Impossible-defs
      by (rule oth-class-taut-3-e[equiv-lr])
    hence [□(p) ∨ □(p⁻) in v]
      apply -
      apply (PLM-subst-goal-method  λφ . □φ ∨ □(p⁻) ¬(p⁻))
      using thm-relation-negation-4 by auto
    hence [□(p) ∨ □(¬p) in v]
      apply -
      apply (PLM-subst-method p⁻ ¬p)
      using thm-relation-negation-3 by auto
    thus [NonContingent p in v]
      unfolding NonContingent-def Necessary-defs Impossible-defs .
  qed

lemma thm-cont-propos-2[PLM]:
  [Contingent p ≡ ◇p & ◇(¬p) in v]
  proof (rule ≡I; rule CP)
    assume [Contingent p in v]
    hence [¬(□p ∨ □(¬p)) in v]
      unfolding Contingent-def Necessary-defs Impossible-defs .
    hence [(¬□p) & (¬□(¬p)) in v]
      by (rule oth-class-taut-6-d[equiv-lr])
    hence [(◇¬(¬p)) & (◇¬p) in v]
      using KBasic2-2[equiv-lr] &I &E by meson
    thus [(◇p) & (◇(¬p)) in v]
      apply - apply PLM-solver
      apply (PLM-subst-method ¬¬p p)
      using oth-class-taut-4-b[equiv-sym] by auto
  next
    assume [(◇p) & (◇¬(p)) in v]
    hence [(◇¬(¬p)) & (◇¬(p)) in v]
      apply - apply PLM-solver
      apply (PLM-subst-method p ¬¬p)
      using oth-class-taut-4-b by auto
    hence [(¬□p) & (¬□(¬p)) in v]
```

```
      using KBasic2-2[equiv-rl] &I &E by meson
    hence [¬(□(p) ∨ □(¬p)) in v]
      by (rule oth-class-taut-6-d[equiv-rl])
    thus [Contingent p in v]
      unfolding Contingent-def Necessary-defs Impossible-defs .
  qed

lemma thm-cont-propos-3[PLM]:
  [Contingent (p::o) ≡ Contingent (p⁻) in v]
  using thm-cont-propos-1
  unfolding NonContingent-def Contingent-def
  by (rule oth-class-taut-5-d[equiv-lr])

context
begin
  private definition p₀ where
    p₀ ≡ ∀ x. ⦇E!,x^P⦈ → ⦇E!,x^P⦈

  lemma thm-noncont-propos-1[PLM]:
    [Necessary p₀ in v]
    unfolding Necessary-defs p₀-def
    apply (rule RN, rule ∀ I)
    using if-p-then-p .

  lemma thm-noncont-propos-2[PLM]:
    [Impossible (p₀⁻) in v]
    unfolding Impossible-defs
    apply (PLM-subst-method ¬p₀ p₀⁻)
     using thm-relation-negation-3[equiv-sym] apply simp
    apply (PLM-subst-method p₀ ¬¬p₀)
     using oth-class-taut-4-b apply simp
    using thm-noncont-propos-1 unfolding Necessary-defs
    by simp

  lemma thm-noncont-propos-3[PLM]:
    [NonContingent (p₀) in v]
    unfolding NonContingent-def using thm-noncont-propos-1
    by (rule ∨I(1))

  lemma thm-noncont-propos-4[PLM]:
    [NonContingent (p₀⁻) in v]
    unfolding NonContingent-def using thm-noncont-propos-2
    by (rule ∨I(2))

  lemma thm-noncont-propos-5[PLM]:
    [∃ (p::o) q . p ≠ q & NonContingent p & NonContingent q in v]
    apply (rule-tac α=p₀ in ∃ I, rule-tac α=p₀⁻ in ∃ I)
    using ∃ I thm-relation-negation-6 thm-noncont-propos-3
          thm-noncont-propos-4 &I by simp

  private definition q₀ where
    q₀ ≡ ∃ x . ⦇E!,x^P⦈ & ◇(¬⦇E!,x^P⦈)

  lemma basic-prop-1[PLM]:
    [∃ p . ◇p & ◇(¬p) in v]
    apply (rule-tac α=q₀ in ∃ I) unfolding q₀-def
    using qml-4[axiom-instance] by simp

  lemma basic-prop-2[PLM]:
    [Contingent q₀ in v]
    unfolding Contingent-def Necessary-defs Impossible-defs
    apply (rule oth-class-taut-6-d[equiv-rl])
    apply (PLM-subst-goal-method λ φ . (¬□(φ)) & ¬□¬q₀ ¬¬q₀)
```

using *oth-class-taut-4-b*[*equiv-sym*] **apply** *simp*
using *qml-4*[*axiom-instance*,*conj-sym*]
unfolding q_0*-def diamond-def* **by** *simp*

lemma *basic-prop-3*[*PLM*]:
[*Contingent* $(q_0{}^-)$ *in v*]
apply (*rule thm-cont-propos-3*[*equiv-lr*])
using *basic-prop-2* **.**

lemma *basic-prop-4*[*PLM*]:
[$\exists$ (p::o) q . $p \neq q$ **&** *Contingent* p **&** *Contingent* q *in v*]
apply (*rule-tac* $\alpha=q_0$ **in** $\exists I$, *rule-tac* $\alpha=q_0{}^-$ **in** $\exists I$)
using *thm-relation-negation-6 basic-prop-2 basic-prop-3* **&***I* **by** *simp*

lemma *four-distinct-props-1*[*PLM*]:
[*NonContingent* $(p::\Pi_0) \rightarrow (\neg(\exists\ q$. *Contingent* q **&** $q = p$)) *in v*]
proof (*rule CP*)
assume [*NonContingent p in v*]
hence [$\neg$(*Contingent p*) *in v*]
unfolding *NonContingent-def Contingent-def*
apply – **by** *PLM-solver*
moreover {
assume [$\exists$ q . *Contingent* q **&** $q = p$ *in v*]
then obtain r **where** [*Contingent* r **&** $r = p$ *in v*]
by (*rule* $\exists E$)
hence [*Contingent p in v*]
using **&***E l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *blast*
}
ultimately show [$\neg(\exists q$. *Contingent* q **&** $q = p$) *in v*]
using *modus-tollens-1 CP* **by** *blast*
qed

lemma *four-distinct-props-2*[*PLM*]:
[*Contingent* (p::o) $\rightarrow \neg(\exists\ q$. (*NonContingent* q **&** $q = p$)) *in v*]
proof (*rule CP*)
assume [*Contingent p in v*]
hence [$\neg$(*NonContingent p*) *in v*]
unfolding *NonContingent-def Contingent-def*
apply – **by** *PLM-solver*
moreover {
assume [$\exists$ q . *NonContingent* q **&** $q = p$ *in v*]
then obtain r **where** [*NonContingent* r **&** $r = p$ *in v*]
by (*rule* $\exists E$)
hence [*NonContingent p in v*]
using **&***E l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *blast*
}
ultimately show [$\neg(\exists q$. *NonContingent* q **&** $q = p$) *in v*]
using *modus-tollens-1 CP* **by** *blast*
qed

lemma *four-distinct-props-4*[*PLM*]:
[$p_0 \neq (p_0{}^-)$ **&** $p_0 \neq q_0$ **&** $p_0 \neq (q_0{}^-)$ **&** $(p_0{}^-) \neq q_0$
& $(p_0{}^-) \neq (q_0{}^-)$ **&** $q_0 \neq (q_0{}^-)$ *in v*]
proof (*rule* **&***I*)+
show [$p_0 \neq (p_0{}^-)$ *in v*]
by (*rule thm-relation-negation-6*)
next
{
assume [$p_0 = q_0$ *in v*]
hence [$\exists$ q . *NonContingent* q **&** $q = q_0$ *in v*]
using **&***I thm-noncont-propos-3* $\exists I$[**where** $\alpha=p_0$]

```
        by simp
      }
      thus [p₀ ≠ q₀ in v]
        using four-distinct-props-2[deduction, OF basic-prop-2]
              modus-tollens-1 CP
        by blast
    next
      {
        assume [p₀ = (q₀⁻) in v]
        hence [∃ q . NonContingent q & q = (q₀⁻) in v]
          using thm-noncont-propos-3 &I ∃I[where α=p₀] by simp
      }
      thus [p₀ ≠ (q₀⁻) in v]
        using four-distinct-props-2[deduction, OF basic-prop-3]
              modus-tollens-1 CP
      by blast
    next
      {
        assume [(p₀⁻) = q₀ in v]
        hence [∃ q . NonContingent q & q = q₀ in v]
          using thm-noncont-propos-4 &I ∃I[where α=p₀⁻] by auto
      }
      thus [(p₀⁻) ≠ q₀ in v]
        using four-distinct-props-2[deduction, OF basic-prop-2]
              modus-tollens-1 CP
        by blast
    next
      {
        assume [(p₀⁻) = (q₀⁻) in v]
        hence [∃ q . NonContingent q & q = (q₀⁻) in v]
          using thm-noncont-propos-4 &I ∃I[where α=p₀⁻] by auto
      }
      thus [(p₀⁻) ≠ (q₀⁻) in v]
        using four-distinct-props-2[deduction, OF basic-prop-3]
              modus-tollens-1 CP
        by blast
    next
      show [q₀ ≠ (q₀⁻) in v]
        by (rule thm-relation-negation-6)
    qed

lemma cont-true-cont-1[PLM]:
  [ContingentlyTrue p → Contingent p in v]
  apply (rule CP, rule thm-cont-propos-2[equiv-rl])
  unfolding ContingentlyTrue-def
  apply (rule &I, drule &E(1))
   using T◇[deduction] apply simp
  by (rule &E(2))

lemma cont-true-cont-2[PLM]:
  [ContingentlyFalse p → Contingent p in v]
  apply (rule CP, rule thm-cont-propos-2[equiv-rl])
  unfolding ContingentlyFalse-def
  apply (rule &I, drule &E(2))
   apply simp
  apply (drule &E(1))
  using T◇[deduction] by simp

lemma cont-true-cont-3[PLM]:
  [ContingentlyTrue p ≡ ContingentlyFalse (p⁻) in v]
  unfolding ContingentlyTrue-def ContingentlyFalse-def
  apply (PLM-subst-method ¬p p⁻)
   using thm-relation-negation-3[equiv-sym] apply simp
```

```
  apply (PLM-subst-method p ¬¬p)
  by PLM-solver+

lemma cont-true-cont-4[PLM]:
  [ContingentlyFalse p ≡ ContingentlyTrue (p⁻) in v]
  unfolding ContingentlyTrue-def ContingentlyFalse-def
  apply (PLM-subst-method ¬p p⁻)
   using thm-relation-negation-3[equiv-sym] apply simp
  apply (PLM-subst-method p ¬¬p)
  by PLM-solver+

lemma cont-tf-thm-1[PLM]:
  [ContingentlyTrue q0 ∨ ContingentlyFalse q0 in v]
  proof −
    have [q0 ∨ ¬q0 in v]
      by PLM-solver
    moreover {
      assume [q0 in v]
      hence [q0 & ◇¬q0 in v]
        unfolding q0-def
        using qml-4[axiom-instance,conj2] &I
        by auto
    }
    moreover {
      assume [¬q0 in v]
      hence [(¬q0) & ◇q0 in v]
        unfolding q0-def
        using qml-4[axiom-instance,conj1] &I
        by auto
    }
    ultimately show ?thesis
      unfolding ContingentlyTrue-def ContingentlyFalse-def
      using ∨E(4) CP by auto
  qed

lemma cont-tf-thm-2[PLM]:
  [ContingentlyFalse q0 ∨ ContingentlyFalse (q0⁻) in v]
  using cont-tf-thm-1 cont-true-cont-3[where p=q0]
        cont-true-cont-4[where p=q0]
  apply − by PLM-solver

lemma cont-tf-thm-3[PLM]:
  [∃ p . ContingentlyTrue p in v]
  proof (rule ∨E(1); (rule CP)?)
    show [ContingentlyTrue q0 ∨ ContingentlyFalse q0 in v]
      using cont-tf-thm-1 .
  next
    assume [ContingentlyTrue q0 in v]
    thus ?thesis
      using ∃ I by metis
  next
    assume [ContingentlyFalse q0 in v]
    hence [ContingentlyTrue (q0⁻) in v]
      using cont-true-cont-4[equiv-lr] by simp
    thus ?thesis
      using ∃ I by metis
  qed

lemma cont-tf-thm-4[PLM]:
  [∃ p . ContingentlyFalse p in v]
  proof (rule ∨E(1); (rule CP)?)
    show [ContingentlyTrue q0 ∨ ContingentlyFalse q0 in v]
      using cont-tf-thm-1 .
```

```
    next
      assume [ContingentlyTrue q₀ in v]
      hence [ContingentlyFalse (q₀⁻) in v]
        using cont-true-cont-3[equiv-lr] by simp
      thus ?thesis
        using ∃ I by metis
    next
      assume [ContingentlyFalse q₀ in v]
      thus ?thesis
        using ∃ I by metis
    qed

  lemma cont-tf-thm-5[PLM]:
    [ContingentlyTrue p & Necessary q → p ≠ q in v]
    proof (rule CP)
      assume [ContingentlyTrue p & Necessary q in v]
      hence 1: [◇(¬p) & □ q in v]
        unfolding ContingentlyTrue-def Necessary-defs
        using &E &I by blast
      hence [¬□p in v]
        apply − apply (drule &E(1))
        unfolding diamond-def
        apply (PLM-subst-method ¬¬p p)
        using oth-class-taut-4-b[equiv-sym] by auto
      moreover {
        assume [p = q in v]
        hence [□p in v]
          using l-identity[where α=q and β=p and φ=λ x . □ x,
                           axiom-instance, deduction, deduction]
                1[conj2] id-eq-prop-prop-8-b[deduction]
          by blast
      }
      ultimately show [p ≠ q in v]
        using modus-tollens-1 CP by blast
    qed

  lemma cont-tf-thm-6[PLM]:
    [(ContingentlyFalse p & Impossible q) → p ≠ q in v]
    proof (rule CP)
      assume [ContingentlyFalse p & Impossible q in v]
      hence 1: [◇p & □(¬q) in v]
        unfolding ContingentlyFalse-def Impossible-defs
        using &E &I by blast
      hence [¬◇q in v]
        unfolding diamond-def apply − by PLM-solver
      moreover {
        assume [p = q in v]
        hence [◇q in v]
          using l-identity[axiom-instance, deduction, deduction] 1[conj1]
                id-eq-prop-prop-8-b[deduction]
          by blast
      }
      ultimately show [p ≠ q in v]
        using modus-tollens-1 CP by blast
    qed
end

lemma oa-contingent-1[PLM]:
  [O! ≠ A! in v]
  proof −
    {
      assume [O! = A! in v]
      hence [(λx. ◇⦇E!,x^P⦈) = (λx. ¬◇⦇E!,x^P⦈) in v]
```

unfolding *Ordinary-def Abstract-def* **.**
moreover have $[\llparenthesis(\boldsymbol{\lambda}x.\ \Diamond\llparenthesis E!,x^P\rrparenthesis),\ x^P\rrparenthesis \equiv \Diamond\llparenthesis E!,x^P\rrparenthesis\ in\ v]$
apply (*rule beta-C-meta-1*)
by *show-proper*
ultimately have $[\llparenthesis(\boldsymbol{\lambda}x.\ \neg\Diamond\llparenthesis E!,x^P\rrparenthesis),\ x^P\rrparenthesis \equiv \Diamond\llparenthesis E!,x^P\rrparenthesis\ in\ v]$
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
moreover have $[\llparenthesis(\boldsymbol{\lambda}x.\ \neg\Diamond\llparenthesis E!,x^P\rrparenthesis),\ x^P\rrparenthesis \equiv \neg\Diamond\llparenthesis E!,x^P\rrparenthesis\ in\ v]$
apply (*rule beta-C-meta-1*)
by *show-proper*
ultimately have $[\Diamond\llparenthesis E!,x^P\rrparenthesis \equiv \neg\Diamond\llparenthesis E!,x^P\rrparenthesis\ in\ v]$
apply − **by** *PLM-solver*
}
thus *?thesis*
using *oth-class-taut-1-b modus-tollens-1 CP*
by *blast*
qed

lemma *oa-contingent-2*[*PLM*]:
$[\llparenthesis O!,x^P\rrparenthesis \equiv \neg\llparenthesis A!,x^P\rrparenthesis\ in\ v]$
proof −
have $[\llparenthesis(\boldsymbol{\lambda}x.\ \neg\Diamond\llparenthesis E!,x^P\rrparenthesis),\ x^P\rrparenthesis \equiv \neg\Diamond\llparenthesis E!,x^P\rrparenthesis\ in\ v]$
apply (*rule beta-C-meta-1*)
by *show-proper*
hence $[(\neg\llparenthesis(\boldsymbol{\lambda}x.\ \neg\Diamond\llparenthesis E!,x^P\rrparenthesis),\ x^P\rrparenthesis) \equiv \Diamond\llparenthesis E!,x^P\rrparenthesis\ in\ v]$
using *oth-class-taut-5-d*[*equiv-lr*] *oth-class-taut-4-b*[*equiv-sym*]
$\equiv E(5)$ **by** *blast*
moreover have $[\llparenthesis(\boldsymbol{\lambda}x.\ \Diamond\llparenthesis E!,x^P\rrparenthesis),\ x^P\rrparenthesis \equiv \Diamond\llparenthesis E!,x^P\rrparenthesis\ in\ v]$
apply (*rule beta-C-meta-1*)
by *show-proper*
ultimately show *?thesis*
unfolding *Ordinary-def Abstract-def*
apply − **by** *PLM-solver*
qed

lemma *oa-contingent-3*[*PLM*]:
$[\llparenthesis A!,x^P\rrparenthesis \equiv \neg\llparenthesis O!,x^P\rrparenthesis\ in\ v]$
using *oa-contingent-2*
apply − **by** *PLM-solver*

lemma *oa-contingent-4*[*PLM*]:
[*Contingent O*! *in v*]
apply (*rule thm-cont-prop-2*[*equiv-rl*], *rule* **&***I*)
subgoal
unfolding *Ordinary-def*
apply (*PLM-subst-method* $\lambda\ x\ .\ \Diamond\llparenthesis E!,x^P\rrparenthesis\ \lambda\ x\ .\ \llparenthesis\boldsymbol{\lambda}x.\ \Diamond\llparenthesis E!,x^P\rrparenthesis,x^P\rrparenthesis$)
apply (*safe intro*!: *beta-C-meta-1*[*equiv-sym*])
apply *show-proper*
using *BF*$\Diamond$[*deduction*, *OF thm-cont-prop-2*[*equiv-lr*, *OF thm-cont-e-2*, *conj1*]]
by (*rule T*$\Diamond$[*deduction*])
subgoal
apply (*PLM-subst-method* $\lambda\ x\ .\ \llparenthesis A!,x^P\rrparenthesis\ \lambda\ x\ .\ \neg\llparenthesis O!,x^P\rrparenthesis$)
using *oa-contingent-3* **apply** *simp*
using *cqt-further-5*[*deduction*,*conj1*, *OF A-objects*[*axiom-instance*]]
by (*rule T*$\Diamond$[*deduction*])
done

lemma *oa-contingent-5*[*PLM*]:
[*Contingent A*! *in v*]
apply (*rule thm-cont-prop-2*[*equiv-rl*], *rule* **&***I*)
subgoal
using *cqt-further-5*[*deduction*,*conj1*, *OF A-objects*[*axiom-instance*]]
by (*rule T*$\Diamond$[*deduction*])
subgoal

```
    unfolding Abstract-def
    apply (PLM-subst-method λ x . ¬◇⦇E!,x^P⦈ λ x . ⦇λx. ¬◇⦇E!,x^P⦈,x^P⦈)
     apply (safe intro!: beta-C-meta-1[equiv-sym])
      apply show-proper
    apply (PLM-subst-method λ x . ◇⦇E!,x^P⦈ λ x . ¬¬◇⦇E!,x^P⦈)
     using oth-class-taut-4-b apply simp
    using BF◇[deduction, OF thm-cont-prop-2[equiv-lr, OF thm-cont-e-2, conj1]]
    by (rule T◇[deduction])
  done

lemma oa-contingent-6[PLM]:
  [(O!⁻) ≠ (A!⁻) in v]
  proof −
    {
      assume [(O!⁻) = (A!⁻) in v]
      hence [(λx. ¬⦇O!,x^P⦈) = (λx. ¬⦇A!,x^P⦈) in v]
        unfolding propnot-defs .
      moreover have [⦇(λx. ¬⦇O!,x^P⦈), x^P⦈ ≡ ¬⦇O!,x^P⦈ in v]
        apply (rule beta-C-meta-1)
        by show-proper
      ultimately have [⦇λx. ¬⦇A!,x^P⦈,x^P⦈ ≡  ¬⦇O!,x^P⦈ in v]
        using l-identity[axiom-instance, deduction, deduction]
        by fast
      hence [(¬⦇A!,x^P⦈) ≡ ¬⦇O!,x^P⦈ in v]
        apply −
        apply (PLM-subst-method ⦇λx. ¬⦇A!,x^P⦈,x^P⦈ (¬⦇A!,x^P⦈))
         apply (safe intro!: beta-C-meta-1)
        by show-proper
      hence [⦇O!,x^P⦈ ≡ ¬⦇O!,x^P⦈ in v]
        using oa-contingent-2 apply − by PLM-solver
    }
    thus ?thesis
      using oth-class-taut-1-b modus-tollens-1 CP
      by blast
  qed

lemma oa-contingent-7[PLM]:
  [⦇O!⁻,x^P⦈ ≡ ¬⦇A!⁻,x^P⦈ in v]
  proof −
    have [(¬⦇λx. ¬⦇A!,x^P⦈,x^P⦈) ≡ ⦇A!,x^P⦈ in v]
      apply (PLM-subst-method (¬⦇A!,x^P⦈) ⦇λx. ¬⦇A!,x^P⦈,x^P⦈)
       apply (safe intro!: beta-C-meta-1[equiv-sym])
        apply show-proper
      using oth-class-taut-4-b[equiv-sym] by auto
    moreover have [⦇λx. ¬⦇O!,x^P⦈,x^P⦈ ≡ ¬⦇O!,x^P⦈ in v]
      apply (rule beta-C-meta-1)
      by show-proper
    ultimately show ?thesis
      unfolding propnot-defs
      using oa-contingent-3
      apply − by PLM-solver
  qed

lemma oa-contingent-8[PLM]:
  [Contingent (O!⁻) in v]
  using oa-contingent-4 thm-cont-prop-3[equiv-lr] by auto

lemma oa-contingent-9[PLM]:
  [Contingent (A!⁻) in v]
  using oa-contingent-5 thm-cont-prop-3[equiv-lr] by auto

lemma oa-facts-1[PLM]:
  [⦇O!,x^P⦈ → □⦇O!,x^P⦈ in v]
```

proof (*rule CP*)
  assume [$\lbrace O!, x^P \rbrace$ *in v*]
  hence [$\Diamond\lbrace E!, x^P \rbrace$ *in v*]
    unfolding *Ordinary-def* **apply** −
    apply (*rule beta-C-meta-1*[*equiv-lr*])
    by *show-proper*
  hence [$\Box\Diamond\lbrace E!, x^P \rbrace$ *in v*]
    using *qml-3*[*axiom-instance*, *deduction*] **by** *auto*
  thus [$\Box\lbrace O!, x^P \rbrace$ *in v*]
    unfolding *Ordinary-def*
    apply −
    apply (*PLM-subst-method* $\Diamond\lbrace E!, x^P \rbrace$ $\lbrace \boldsymbol{\lambda} x.\ \Diamond\lbrace E!, x^P \rbrace, x^P \rbrace$)
     apply (*safe intro*!: *beta-C-meta-1*[*equiv-sym*])
    by *show-proper*
qed

lemma *oa-facts-2*[*PLM*]:
[$\lbrace A!, x^P \rbrace \rightarrow \Box\lbrace A!, x^P \rbrace$ *in v*]
proof (*rule CP*)
  assume [$\lbrace A!, x^P \rbrace$ *in v*]
  hence [$\neg\Diamond\lbrace E!, x^P \rbrace$ *in v*]
    unfolding *Abstract-def* **apply** −
    apply (*rule beta-C-meta-1*[*equiv-lr*])
    by *show-proper*
  hence [$\Box\Box\neg\lbrace E!, x^P \rbrace$ *in v*]
    using *KBasic2-4*[*equiv-rl*] *4*$\Box$[*deduction*] **by** *auto*
  hence [$\Box\neg\Diamond\lbrace E!, x^P \rbrace$ *in v*]
    apply −
    apply (*PLM-subst-method* $\Box\neg\lbrace E!, x^P \rbrace$ $\neg\Diamond\lbrace E!, x^P \rbrace$)
    using *KBasic2-4* **by** *auto*
  thus [$\Box\lbrace A!, x^P \rbrace$ *in v*]
    unfolding *Abstract-def*
    apply −
    apply (*PLM-subst-method* $\neg\Diamond\lbrace E!, x^P \rbrace$ $\lbrace \boldsymbol{\lambda} x.\ \neg\Diamond\lbrace E!, x^P \rbrace, x^P \rbrace$)
     apply (*safe intro*!: *beta-C-meta-1*[*equiv-sym*])
    by *show-proper*
qed

lemma *oa-facts-3*[*PLM*]:
[$\Diamond\lbrace O!, x^P \rbrace \rightarrow \lbrace O!, x^P \rbrace$ *in v*]
using *oa-facts-1* **by** (*rule derived-S5-rules-2-b*)

lemma *oa-facts-4*[*PLM*]:
[$\Diamond\lbrace A!, x^P \rbrace \rightarrow \lbrace A!, x^P \rbrace$ *in v*]
using *oa-facts-2* **by** (*rule derived-S5-rules-2-b*)

lemma *oa-facts-5*[*PLM*]:
[$\Diamond\lbrace O!, x^P \rbrace \equiv \Box\lbrace O!, x^P \rbrace$ *in v*]
using *oa-facts-1*[*deduction*, *OF oa-facts-3*[*deduction*]]
  T$\Diamond$[*deduction*, *OF qml-2*[*axiom-instance*, *deduction*]]
  $\equiv$*I CP* **by** *blast*

lemma *oa-facts-6*[*PLM*]:
[$\Diamond\lbrace A!, x^P \rbrace \equiv \Box\lbrace A!, x^P \rbrace$ *in v*]
using *oa-facts-2*[*deduction*, *OF oa-facts-4*[*deduction*]]
  T$\Diamond$[*deduction*, *OF qml-2*[*axiom-instance*, *deduction*]]
  $\equiv$*I CP* **by** *blast*

lemma *oa-facts-7*[*PLM*]:
[$\lbrace O!, x^P \rbrace \equiv \boldsymbol{\mathcal{A}}\lbrace O!, x^P \rbrace$ *in v*]
apply (*rule* $\equiv$*I*; *rule CP*)
 apply (*rule nec-imp-act*[*deduction*, *OF oa-facts-1*[*deduction*]]; *assumption*)
proof −

```
    assume [𝓐⦇O!,x^P⦈ in v]
    hence [𝓐(◇⦇E!,x^P⦈) in v]
      unfolding Ordinary-def apply −
      apply (PLM-subst-method ⦇λx. ◇⦇E!,x^P⦈,x^P⦈ ◇⦇E!,x^P⦈)
      apply (safe intro!: beta-C-meta-1)
      by show-proper
    hence [◇⦇E!,x^P⦈ in v]
      using Act-Basic-6[equiv-rl] by auto
    thus [⦇O!,x^P⦈ in v]
      unfolding Ordinary-def apply −
      apply (PLM-subst-method ◇⦇E!,x^P⦈ ⦇λx. ◇⦇E!,x^P⦈,x^P⦈)
       apply (safe intro!: beta-C-meta-1[equiv-sym])
      by show-proper
  qed

lemma oa-facts-8[PLM]:
  [⦇A!,x^P⦈ ≡ 𝓐⦇A!,x^P⦈ in v]
  apply (rule ≡I; rule CP)
   apply (rule nec-imp-act[deduction, OF oa-facts-2[deduction]]; assumption)
  proof −
    assume [𝓐⦇A!,x^P⦈ in v]
    hence [𝓐(¬◇⦇E!,x^P⦈) in v]
      unfolding Abstract-def apply −
      apply (PLM-subst-method ⦇λx. ¬◇⦇E!,x^P⦈,x^P⦈ ¬◇⦇E!,x^P⦈)
      apply (safe intro!: beta-C-meta-1)
      by show-proper
    hence [𝓐(□¬⦇E!,x^P⦈) in v]
      apply −
      apply (PLM-subst-method (¬◇⦇E!,x^P⦈) (□¬⦇E!,x^P⦈))
      using KBasic2-4[equiv-sym] by auto
    hence [¬◇⦇E!,x^P⦈ in v]
      using qml-act-2[axiom-instance, equiv-rl] KBasic2-4[equiv-lr] by auto
    thus [⦇A!,x^P⦈ in v]
      unfolding Abstract-def apply −
      apply (PLM-subst-method ¬◇⦇E!,x^P⦈ ⦇λx. ¬◇⦇E!,x^P⦈,x^P⦈)
      apply (safe intro!: beta-C-meta-1[equiv-sym])
      by show-proper
  qed

lemma cont-nec-fact1-1[PLM]:
  [WeaklyContingent F ≡ WeaklyContingent (F⁻) in v]
  proof (rule ≡I; rule CP)
    assume [WeaklyContingent F in v]
    hence wc-def: [Contingent F & (∀ x . (◇⦇F,x^P⦈ → □⦇F,x^P⦈)) in v]
      unfolding WeaklyContingent-def .
    have [Contingent (F⁻) in v]
      using wc-def[conj1] by (rule thm-cont-prop-3[equiv-lr])
    moreover {
      {
        fix x
        assume [◇⦇F⁻,x^P⦈ in v]
        hence [¬□⦇F,x^P⦈ in v]
          unfolding diamond-def apply −
          apply (PLM-subst-method ¬⦇F⁻,x^P⦈ ⦇F,x^P⦈)
           using thm-relation-negation-2-1 by auto
        moreover {
          assume [¬□⦇F⁻,x^P⦈ in v]
          hence [¬□⦇λx. ¬⦇F,x^P⦈,x^P⦈ in v]
            unfolding propnot-defs .
          hence [◇⦇F,x^P⦈ in v]
            unfolding diamond-def
            apply − apply (PLM-subst-method ⦇λx. ¬⦇F,x^P⦈,x^P⦈ ¬⦇F,x^P⦈)
            apply (safe intro!: beta-C-meta-1)
```

```
          by show-proper
        hence [□⦇F,x^P⦈ in v]
          using wc-def[conj2] cqt-1[axiom-instance, deduction]
                modus-ponens by fast
      }
      ultimately have [□⦇F⁻, x^P⦈ in v]
        using ¬¬E modus-tollens-1 CP by blast
    }
    hence [∀ x . ◇⦇F⁻,x^P⦈ → □⦇F⁻, x^P⦈ in v]
      using ∀I CP by fast
  }
  ultimately show [WeaklyContingent (F⁻) in v]
    unfolding WeaklyContingent-def by (rule &I)
next
  assume [WeaklyContingent (F⁻) in v]
  hence wc-def: [Contingent (F⁻) & (∀ x . (◇⦇F⁻,x^P⦈ → □⦇F⁻,x^P⦈)) in v]
    unfolding WeaklyContingent-def .
  have [Contingent F in v]
    using wc-def[conj1] by (rule thm-cont-prop-3[equiv-rl])
  moreover {
    {
      fix x
      assume [◇⦇F,x^P⦈ in v]
      hence [¬□⦇F⁻,x^P⦈ in v]
        unfolding diamond-def apply -
        apply (PLM-subst-method ¬⦇F,x^P⦈ ⦇F⁻,x^P⦈)
        using thm-relation-negation-1-1[equiv-sym] by auto
      moreover {
        assume [¬□⦇F,x^P⦈ in v]
        hence [◇⦇F⁻,x^P⦈ in v]
          unfolding diamond-def
          apply - apply (PLM-subst-method ⦇F,x^P⦈ ¬⦇F⁻,x^P⦈)
          using thm-relation-negation-2-1[equiv-sym] by auto
        hence [□⦇F⁻,x^P⦈ in v]
          using wc-def[conj2] cqt-1[axiom-instance, deduction]
                modus-ponens by fast
      }
      ultimately have [□⦇F, x^P⦈ in v]
        using ¬¬E modus-tollens-1 CP by blast
    }
    hence [∀ x . ◇⦇F,x^P⦈ → □⦇F, x^P⦈ in v]
      using ∀I CP by fast
  }
  ultimately show [WeaklyContingent (F) in v]
    unfolding WeaklyContingent-def by (rule &I)
qed

lemma cont-nec-fact1-2[PLM]:
  [(WeaklyContingent F & ¬(WeaklyContingent G)) → (F ≠ G) in v]
  using l-identity[axiom-instance,deduction,deduction] &E &I
      modus-tollens-1 CP by metis

lemma cont-nec-fact2-1[PLM]:
  [WeaklyContingent (O!) in v]
  unfolding WeaklyContingent-def
  apply (rule &I)
   using oa-contingent-4 apply simp
  using oa-facts-5 unfolding equiv-def
  using &E(1) ∀I by fast

lemma cont-nec-fact2-2[PLM]:
  [WeaklyContingent (A!) in v]
  unfolding WeaklyContingent-def
```

```
  apply (rule &I)
   using oa-contingent-5 apply simp
  using oa-facts-6 unfolding equiv-def
  using &E(1) ∀I by fast

lemma cont-nec-fact2-3[PLM]:
  [¬(WeaklyContingent (E!)) in v]
  proof (rule modus-tollens-1, rule CP)
    assume [WeaklyContingent E! in v]
    thus [∀ x . ◇⦇E!,x^P⦈ → □⦇E!,x^P⦈ in v]
    unfolding WeaklyContingent-def using &E(2) by fast
  next
    {
      assume 1: [∀ x . ◇⦇E!,x^P⦈ → □⦇E!,x^P⦈ in v]
      have [∃ x . ◇(⦇E!,x^P⦈ & ◇(¬⦇E!,x^P⦈)) in v]
        using qml-4[axiom-instance,conj1, THEN BFs-3[deduction]] .
      then obtain x where [◇(⦇E!,x^P⦈ & ◇(¬⦇E!,x^P⦈)) in v]
        by (rule ∃E)
      hence [◇⦇E!,x^P⦈ & ◇(¬⦇E!,x^P⦈) in v]
        using KBasic2-8[deduction] S5Basic-8[deduction]
              &I &E by blast
      hence [□⦇E!,x^P⦈ & (¬□⦇E!,x^P⦈) in v]
        using 1[THEN ∀E, deduction] &E &I
              KBasic2-2[equiv-rl] by blast
      hence [¬(∀ x . ◇⦇E!,x^P⦈ → □⦇E!,x^P⦈) in v]
        using oth-class-taut-1-a modus-tollens-1 CP by blast
    }
    thus [¬(∀ x . ◇⦇E!,x^P⦈ → □⦇E!,x^P⦈) in v]
      using reductio-aa-2 if-p-then-p CP by meson
  qed

lemma cont-nec-fact2-4[PLM]:
  [¬(WeaklyContingent (PLM.L)) in v]
  proof -
    {
      assume [WeaklyContingent PLM.L in v]
      hence [Contingent PLM.L in v]
        unfolding WeaklyContingent-def using &E(1) by blast
    }
    thus ?thesis
      using thm-noncont-e-e-3
      unfolding Contingent-def NonContingent-def
      using modus-tollens-2 CP by blast
  qed

lemma cont-nec-fact2-5[PLM]:
  [O! ≠ E! & O! ≠ (E!⁻) & O! ≠ PLM.L & O! ≠ (PLM.L⁻) in v]
  proof ((rule &I)+)
    show [O! ≠ E! in v]
      using cont-nec-fact2-1 cont-nec-fact2-3
            cont-nec-fact1-2[deduction] &I by simp
  next
    have [¬(WeaklyContingent (E!⁻)) in v]
      using cont-nec-fact1-1[THEN oth-class-taut-5-d[equiv-lr], equiv-lr]
            cont-nec-fact2-3 by auto
    thus [O! ≠ (E!⁻) in v]
      using cont-nec-fact2-1 cont-nec-fact1-2[deduction] &I by simp
  next
    show [O! ≠ PLM.L in v]
      using cont-nec-fact2-1 cont-nec-fact2-4
            cont-nec-fact1-2[deduction] &I by simp
  next
    have [¬(WeaklyContingent (PLM.L⁻)) in v]
```

using *cont-nec-fact1-1*[*THEN oth-class-taut-5-d*[*equiv-lr*], *equiv-lr*]
cont-nec-fact2-4 **by** *auto*
thus $[O! \neq (PLM.L^-)\ in\ v]$
using *cont-nec-fact2-1 cont-nec-fact1-2*[*deduction*] **&***I* **by** *simp*
qed

lemma *cont-nec-fact2-6*[*PLM*]:
$[A! \neq E!$ **&** $A! \neq (E!^-)$ **&** $A! \neq PLM.L$ **&** $A! \neq (PLM.L^-)\ in\ v]$
proof ((*rule* **&***I*)+)
show $[A! \neq E!\ in\ v]$
using *cont-nec-fact2-2 cont-nec-fact2-3*
cont-nec-fact1-2[*deduction*] **&***I* **by** *simp*
next
have $[\neg(WeaklyContingent\ (E!^-))\ in\ v]$
using *cont-nec-fact1-1*[*THEN oth-class-taut-5-d*[*equiv-lr*], *equiv-lr*]
cont-nec-fact2-3 **by** *auto*
thus $[A! \neq (E!^-)\ in\ v]$
using *cont-nec-fact2-2 cont-nec-fact1-2*[*deduction*] **&***I* **by** *simp*
next
show $[A! \neq PLM.L\ in\ v]$
using *cont-nec-fact2-2 cont-nec-fact2-4*
cont-nec-fact1-2[*deduction*] **&***I* **by** *simp*
next
have $[\neg(WeaklyContingent\ (PLM.L^-))\ in\ v]$
using *cont-nec-fact1-1*[*THEN oth-class-taut-5-d*[*equiv-lr*],
equiv-lr] *cont-nec-fact2-4* **by** *auto*
thus $[A! \neq (PLM.L^-)\ in\ v]$
using *cont-nec-fact2-2 cont-nec-fact1-2*[*deduction*] **&***I* **by** *simp*
qed

lemma *id-nec3-1*[*PLM*]:
$[((x^P) =_E (y^P)) \equiv (\Box((x^P) =_E (y^P)))\ \ in\ v]$
proof (*rule* $\equiv I$; *rule CP*)
assume $[(x^P) =_E (y^P)\ in\ v]$
hence $[(\!|O!,x^P|\!)\ in\ v] \wedge [(\!|O!,y^P|\!)\ in\ v] \wedge [\Box(\forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!))\ in\ v]$
using *eq-E-simple-1*[*equiv-lr*] **using** **&***E* **by** *blast*
hence $[\Box(\!|O!,x^P|\!)\ in\ v] \wedge [\Box(\!|O!,y^P|\!)\ in\ v]$
$\wedge\ [\Box\Box(\forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!))\ in\ v]$
using *oa-facts-1*[*deduction*] *S5Basic-6*[*deduction*] **by** *blast*
hence $[\Box((\!|O!,x^P|\!)$ **&** $(\!|O!,y^P|\!)$ **&** $\Box(\forall\ F.\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!)))\ in\ v]$
using **&***I KBasic-3*[*equiv-rl*] **by** *presburger*
thus $[\Box((x^P) =_E (y^P))\ in\ v]$
apply −
apply (*PLM-subst-method*
$((\!|O!,x^P|\!)$ **&** $(\!|O!,y^P|\!)$ **&** $\Box(\forall\ F.\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!)))$
$(x^P) =_E (y^P))$
using *eq-E-simple-1*[*equiv-sym*] **by** *auto*
next
assume $[\Box((x^P) =_E (y^P))\ in\ v]$
thus $[((x^P) =_E (y^P))\ in\ v]$
using *qml-2*[*axiom-instance,deduction*] **by** *simp*
qed

lemma *id-nec3-2*[*PLM*]:
$[\Diamond((x^P) =_E (y^P)) \equiv ((x^P) =_E (y^P))\ in\ v]$
proof (*rule* $\equiv I$; *rule CP*)
assume $[\Diamond((x^P) =_E (y^P))\ in\ v]$
thus $[(x^P) =_E (y^P)\ in\ v]$
using *derived-S5-rules-2-b*[*deduction*] *id-nec3-1*[*equiv-lr*]
CP modus-ponens **by** *blast*
next
assume $[(x^P) =_E (y^P)\ in\ v]$
thus $[\Diamond((x^P) =_E (y^P))\ in\ v]$

by (*rule TBasic*[*deduction*])
qed

lemma *thm-neg-eqE*[*PLM*]:
$[((x^P) \neq_E (y^P)) \equiv (\neg((x^P) =_E (y^P)))$ *in v*]
proof −
have $[(x^P) \neq_E (y^P)$ *in v*] = $[(\!|(\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ (x^P) =_E (y^P)))^-, x^P, y^P|\!)$ *in v*]
unfolding *not-identical*$_E$*-def* **by** *simp*
also have ... = $[\neg(\!|(\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ (x^P) =_E (y^P))), x^P, y^P|\!)$ *in v*]
unfolding *propnot-defs*
apply (*safe intro*!: *beta-C-meta-2*[*equiv-lr*] *beta-C-meta-2*[*equiv-rl*])
by *show-proper*+
also have ... = $[\neg((x^P) =_E (y^P))$ *in v*]
apply (*PLM-subst-method*
$(\!|(\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ (x^P) =_E (y^P))), x^P, y^P|\!)$
$(x^P) =_E (y^P))$
apply (*safe intro*!: *beta-C-meta-2*)
unfolding *identity-defs* **by** *show-proper*
finally show *?thesis*
using $\equiv I$ *CP* **by** *presburger*
qed

lemma *id-nec4-1*[*PLM*]:
$[((x^P) \neq_E (y^P)) \equiv \Box((x^P) \neq_E (y^P))$ *in v*]
proof −
have $[(\neg((x^P) =_E (y^P))) \equiv \Box(\neg((x^P) =_E (y^P)))$ *in v*]
using *id-nec3-2*[*equiv-sym*] *oth-class-taut-5-d*[*equiv-lr*]
KBasic2-4[*equiv-sym*] *intro-elim-6-e* **by** *fast*
thus *?thesis*
apply −
apply (*PLM-subst-method* $(\neg((x^P) =_E (y^P)))$ $(x^P) \neq_E (y^P)$)
using *thm-neg-eqE*[*equiv-sym*] **by** *auto*
qed

lemma *id-nec4-2*[*PLM*]:
$[\Diamond((x^P) \neq_E (y^P)) \equiv ((x^P) \neq_E (y^P))$ *in v*]
using $\equiv I$ *id-nec4-1*[*equiv-lr*] *derived-S5-rules-2-b* *CP* $T\Diamond$ **by** *simp*

lemma *id-act-1*[*PLM*]:
$[((x^P) =_E (y^P)) \equiv (\boldsymbol{\mathcal{A}}((x^P) =_E (y^P)))$ *in v*]
proof (*rule* $\equiv I$; *rule CP*)
assume $[(x^P) =_E (y^P)$ *in v*]
hence $[\Box((x^P) =_E (y^P))$ *in v*]
using *id-nec3-1*[*equiv-lr*] **by** *auto*
thus $[\boldsymbol{\mathcal{A}}((x^P) =_E (y^P))$ *in v*]
using *nec-imp-act*[*deduction*] **by** *fast*
next
assume $[\boldsymbol{\mathcal{A}}((x^P) =_E (y^P))$ *in v*]
hence $[\boldsymbol{\mathcal{A}}((\!|O!,x^P|\!)\ \&\ (\!|O!,y^P|\!)\ \&\ \Box(\forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!)))$ *in v*]
apply −
apply (*PLM-subst-method*
$(x^P) =_E (y^P)$
$((\!|O!,x^P|\!)\ \&\ (\!|O!,y^P|\!)\ \&\ \Box(\forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!))))$
using *eq-E-simple-1* **by** *auto*
hence $[\boldsymbol{\mathcal{A}}(\!|O!,x^P|\!)\ \&\ \boldsymbol{\mathcal{A}}(\!|O!,y^P|\!)\ \&\ \boldsymbol{\mathcal{A}}(\Box(\forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!)))$ *in v*]
using *Act-Basic-2*[*equiv-lr*] &*I* &*E* **by** *meson*
thus $[(x^P) =_E (y^P)$ *in v*]
apply − **apply** (*rule eq-E-simple-1*[*equiv-rl*])
using *oa-facts-7*[*equiv-rl*] *qml-act-2*[*axiom-instance*, *equiv-rl*]
&*I* &*E* **by** *meson*
qed

lemma *id-act-2*[*PLM*]:

```
    [((x^P) ≠E (y^P)) ≡ (𝓐((x^P) ≠E (y^P))) in v]
    apply (PLM-subst-method (¬((x^P) =E (y^P))) ((x^P) ≠E (y^P)))
     using thm-neg-eqE[equiv-sym] apply simp
    using id-act-1 oth-class-taut-5-d[equiv-lr] thm-neg-eqE intro-elim-6-e
          logic-actual-nec-1[axiom-instance,equiv-sym] by meson

end

class id-act = id-eq +
  assumes id-act-prop: [𝓐(α = β) in v] ⟹ [(α = β) in v]

instantiation ν :: id-act
begin
  instance proof
    interpret PLM .
    fix x::ν and y::ν and v::i
    assume [𝓐(x = y) in v]
    hence [𝓐(((x^P) =E (y^P)) ∨ ((⦇A!,x^P⦈ & ⦇A!,y^P⦈)
          & □(∀ F . ⦃x^P,F⦄ ≡ ⦃y^P,F⦄))) in v]
      unfolding identity-defs by auto
    hence [𝓐(((x^P) =E (y^P))) ∨ 𝓐(((⦇A!,x^P⦈ & ⦇A!,y^P⦈)
          & □(∀ F . ⦃x^P,F⦄ ≡ ⦃y^P,F⦄))) in v]
      using Act-Basic-10[equiv-lr] by auto
    moreover {
       assume [𝓐(((x^P) =E (y^P))) in v]
       hence [(x^P) = (y^P) in v]
        using id-act-1[equiv-rl] eq-E-simple-2[deduction] by auto
    }
    moreover {
       assume [𝓐(⦇A!,x^P⦈ & ⦇A!,y^P⦈ & □(∀ F . ⦃x^P,F⦄ ≡ ⦃y^P,F⦄)) in v]
       hence [𝓐⦇A!,x^P⦈ & 𝓐⦇A!,y^P⦈ & 𝓐(□(∀ F . ⦃x^P,F⦄ ≡ ⦃y^P,F⦄)) in v]
         using Act-Basic-2[equiv-lr] &I &E by meson
       hence [⦇A!,x^P⦈ & ⦇A!,y^P⦈ & (□(∀ F . ⦃x^P,F⦄ ≡ ⦃y^P,F⦄)) in v]
         using oa-facts-8[equiv-rl] qml-act-2[axiom-instance,equiv-rl]
           &I &E by meson
       hence [(x^P) = (y^P) in v]
        unfolding identity-defs using ∨I by auto
    }
    ultimately have [(x^P) = (y^P) in v]
      using intro-elim-4-a CP by meson
    thus [x = y in v]
      unfolding identity-defs by auto
  qed
end

instantiation Π₁ :: id-act
begin
  instance proof
    interpret PLM .
    fix F::Π₁ and G::Π₁ and v::i
    show [𝓐(F = G) in v] ⟹ [(F = G) in v]
      unfolding identity-defs
      using qml-act-2[axiom-instance,equiv-rl] by auto
  qed
end

instantiation o :: id-act
begin
  instance proof
    interpret PLM .
    fix p :: o and q :: o and v::i
    show [𝓐(p = q) in v] ⟹ [p = q in v]
      unfolding identity_o-def using id-act-prop by blast
```

```
  qed
end

instantiation Π₂ :: id-act
begin
  instance proof
    interpret PLM .
    fix F::Π₂ and G::Π₂ and v::i
    assume a: [𝒜(F = G) in v]
    {
      fix x
      have [𝒜((λy. (|F,x^P,y^P|)) = (λy. (|G,x^P,y^P|))
             & (λy. (|F,y^P,x^P|)) = (λy. (|G,y^P,x^P|))) in v]
        using a logic-actual-nec-3[axiom-instance, equiv-lr] cqt-basic-4[equiv-lr] ∀E
        unfolding identity₂-def by fast
      hence [((λy. (|F,x^P,y^P|)) = (λy. (|G,x^P,y^P|)))
              & ((λy. (|F,y^P,x^P|)) = (λy. (|G,y^P,x^P|))) in v]
        using &I &E id-act-prop Act-Basic-2[equiv-lr] by metis
    }
    thus [F = G in v] unfolding identity-defs by (rule ∀I)
  qed
end

instantiation Π₃ :: id-act
begin
  instance proof
    interpret PLM .
    fix F::Π₃ and G::Π₃ and v::i
    assume a: [𝒜(F = G) in v]
    let ?p = λ x y . (λz. (|F,z^P,x^P,y^P|)) = (λz. (|G,z^P,x^P,y^P|))
                   & (λz. (|F,x^P,z^P,y^P|)) = (λz. (|G,x^P,z^P,y^P|))
                   & (λz. (|F,x^P,y^P,z^P|)) = (λz. (|G,x^P,y^P,z^P|))
    {
      fix x
      {
        fix y
        have [𝒜(?p x y) in v]
          using a logic-actual-nec-3[axiom-instance, equiv-lr]
                cqt-basic-4[equiv-lr] ∀E[where 'a=ν]
          unfolding identity₃-def by blast
        hence [?p x y in v]
          using &I &E id-act-prop Act-Basic-2[equiv-lr] by metis
      }
      hence [∀ y . ?p x y in v]
        by (rule ∀I)
    }
    thus [F = G in v]
      unfolding identity₃-def by (rule ∀I)
  qed
end

context PLM
begin
  lemma id-act-3[PLM]:
    [((α::('a::id-act)) = β) ≡ 𝒜(α = β) in v]
    using ≡I CP id-nec[equiv-lr, THEN nec-imp-act[deduction]]
          id-act-prop by metis

  lemma id-act-4[PLM]:
    [((α::('a::id-act)) ≠ β) ≡ 𝒜(α ≠ β) in v]
    using id-act-3[THEN oth-class-taut-5-d[equiv-lr]]
          logic-actual-nec-1[axiom-instance, equiv-sym]
          intro-elim-6-e by blast
```

lemma *id-act-desc*[*PLM*]:
  $[(y^P) = (\iota x \,.\, x = y) \; in \; v]$
  using *descriptions*[*axiom-instance*,*equiv-rl*]
    id-act-3[*equiv-sym*] $\forall I$ **by** *fast*

lemma *eta-conversion-lemma-1*[*PLM*]:
  $[(\boldsymbol{\lambda} \; x \,.\, (|F,x^P|)) = F \; in \; v]$
  using *lambda-predicates-3-1*[*axiom-instance*] **.**

lemma *eta-conversion-lemma-0*[*PLM*]:
  $[(\boldsymbol{\lambda}^0 \; p) = p \; in \; v]$
  using *lambda-predicates-3-0*[*axiom-instance*] **.**

lemma *eta-conversion-lemma-2*[*PLM*]:
  $[(\boldsymbol{\lambda}^2 \; (\lambda \; x \; y \,.\, (|F,x^P,y^P|))) = F \; in \; v]$
  using *lambda-predicates-3-2*[*axiom-instance*] **.**

lemma *eta-conversion-lemma-3*[*PLM*]:
  $[(\boldsymbol{\lambda}^3 \; (\lambda \; x \; y \; z \,.\, (|F,x^P,y^P,z^P|))) = F \; in \; v]$
  using *lambda-predicates-3-3*[*axiom-instance*] **.**

lemma *lambda-p-q-p-eq-q*[*PLM*]:
  $[((\boldsymbol{\lambda}^0 \; p) = (\boldsymbol{\lambda}^0 \; q)) \equiv (p = q) \; in \; v]$
  using *eta-conversion-lemma-0*
    l-identity[*axiom-instance*, *deduction*, *deduction*]
    eta-conversion-lemma-0[*eq-sym*] $\equiv I$ *CP*
  by *metis*

9.12 The Theory of Objects

lemma *partition-1*[*PLM*]:
  $[\forall \; x \,.\, (|O!,x^P|) \lor (|A!,x^P|) \; in \; v]$
  proof (*rule* $\forall I$)
    fix x
    have $[\Diamond(|E!,x^P|) \lor \neg\Diamond(|E!,x^P|) \; in \; v]$
      by *PLM-solver*
    moreover have $[\Diamond(|E!,x^P|) \equiv (|\boldsymbol{\lambda} \; y \,.\, \Diamond(|E!,y^P|), \; x^P|) \; in \; v]$
      apply (*rule beta-C-meta-1*[*equiv-sym*])
      by *show-proper*
    moreover have $[(\neg\Diamond(|E!,x^P|)) \equiv (|\boldsymbol{\lambda} \; y \,.\, \neg\Diamond(|E!,y^P|), \; x^P|) \; in \; v]$
      apply (*rule beta-C-meta-1*[*equiv-sym*])
      by *show-proper*
    ultimately show $[(|O!, \; x^P|) \lor (|A!, \; x^P|) \; in \; v]$
      unfolding *Ordinary-def Abstract-def* **by** *PLM-solver*
  qed

lemma *partition-2*[*PLM*]:
  $[\neg(\exists \; x \,.\, (|O!,x^P|) \; \& \; (|A!,x^P|)) \; in \; v]$
  proof −
    {
      assume $[\exists \; x \,.\, (|O!,x^P|) \; \& \; (|A!,x^P|) \; in \; v]$
      then obtain b **where** $[(|O!,b^P|) \; \& \; (|A!,b^P|) \; in \; v]$
        by (*rule* $\exists E$)
      hence *?thesis*
        using **&***E oa-contingent-2*[*equiv-lr*]
          reductio-aa-2 **by** *fast*
    }
    thus *?thesis*
      using *reductio-aa-2* **by** *blast*
  qed

lemma *ord-eq-Eequiv-1*[*PLM*]:

$[\llparenthesis O!,x\rrparenthesis \rightarrow (x =_E x)\ in\ v]$
proof (*rule CP*)
assume $[\llparenthesis O!,x\rrparenthesis\ in\ v]$
moreover have $[\Box(\forall\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,x\rrparenthesis)\ in\ v]$
by *PLM-solver*
ultimately show $[(x) =_E (x)\ in\ v]$
using **&***I* *eq-E-simple-1*[*equiv-rl*] **by** *blast*
qed

lemma *ord-eq-Eequiv-2*[*PLM*]:
$[(x =_E y) \rightarrow (y =_E x)\ in\ v]$
proof (*rule CP*)
assume $[x =_E y\ in\ v]$
hence *1*: $[\llparenthesis O!,x\rrparenthesis$ **&** $\llparenthesis O!,y\rrparenthesis$ **&** $\Box(\forall\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,y\rrparenthesis)\ in\ v]$
using *eq-E-simple-1*[*equiv-lr*] **by** *simp*
have $[\Box(\forall\ F\ .\ \llparenthesis F,y\rrparenthesis \equiv \llparenthesis F,x\rrparenthesis)\ in\ v]$
apply (*PLM-subst-method*
$\lambda\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,y\rrparenthesis$
$\lambda\ F\ .\ \llparenthesis F,y\rrparenthesis \equiv \llparenthesis F,x\rrparenthesis$)
using *oth-class-taut-3-g 1*[*conj2*] **by** *auto*
thus $[y =_E x\ in\ v]$
using *eq-E-simple-1*[*equiv-rl*] *1*[*conj1*]
&*E* **&***I* **by** *meson*
qed

lemma *ord-eq-Eequiv-3*[*PLM*]:
$[((x =_E y)$ **&** $(y =_E z)) \rightarrow (x =_E z)\ in\ v]$
proof (*rule CP*)
assume *a*: $[(x =_E y)$ **&** $(y =_E z)\ in\ v]$
have $[\Box((\forall\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,y\rrparenthesis)$ **&** $(\forall\ F\ .\ \llparenthesis F,y\rrparenthesis \equiv \llparenthesis F,z\rrparenthesis))\ in\ v]$
using *KBasic-3*[*equiv-rl*] *a*[*conj1*, *THEN eq-E-simple-1*[*equiv-lr*,*conj2*]]
a[*conj2*, *THEN eq-E-simple-1*[*equiv-lr*,*conj2*]] **&***I* **by** *blast*
moreover {
{
fix *w*
have $[((\forall\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,y\rrparenthesis)$ **&** $(\forall\ F\ .\ \llparenthesis F,y\rrparenthesis \equiv \llparenthesis F,z\rrparenthesis))$
$\rightarrow (\forall\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,z\rrparenthesis)\ in\ w]$
by *PLM-solver*
}
hence $[\Box(((\forall\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,y\rrparenthesis)$ **&** $(\forall\ F\ .\ \llparenthesis F,y\rrparenthesis \equiv \llparenthesis F,z\rrparenthesis))$
$\rightarrow (\forall\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,z\rrparenthesis))\ in\ v]$
by (*rule RN*)
}
ultimately have $[\Box(\forall\ F\ .\ \llparenthesis F,x\rrparenthesis \equiv \llparenthesis F,z\rrparenthesis)\ in\ v]$
using *qml-1*[*axiom-instance*,*deduction*,*deduction*] **by** *blast*
thus $[x =_E z\ in\ v]$
using *a*[*conj1*, *THEN eq-E-simple-1*[*equiv-lr*,*conj1*,*conj1*]]
using *a*[*conj2*, *THEN eq-E-simple-1*[*equiv-lr*,*conj1*,*conj2*]]
eq-E-simple-1[*equiv-rl*] **&***I*
by *presburger*
qed

lemma *ord-eq-E-eq*[*PLM*]:
$[(\llparenthesis O!,x^P\rrparenthesis \vee \llparenthesis O!,y^P\rrparenthesis) \rightarrow ((x^P = y^P) \equiv (x^P =_E y^P))\ in\ v]$
proof (*rule CP*)
assume $[\llparenthesis O!,x^P\rrparenthesis \vee \llparenthesis O!,y^P\rrparenthesis\ in\ v]$
moreover {
assume $[\llparenthesis O!,x^P\rrparenthesis\ in\ v]$
hence $[(x^P = y^P) \equiv (x^P =_E y^P)\ in\ v]$
using $\equiv$*I CP l-identity*[*axiom-instance*, *deduction*, *deduction*]
ord-eq-Eequiv-1[*deduction*] *eq-E-simple-2*[*deduction*] **by** *metis*
}
moreover {

assume $[(\!| O!,y^P |\!) \ in \ v]$
hence $[(x^P = y^P) \equiv (x^P =_E y^P) \ in \ v]$
using $\equiv I$ *CP l-identity*[*axiom-instance*, *deduction*, *deduction*]
ord-eq-Eequiv-1[*deduction*] *eq-E-simple-2*[*deduction*] *id-eq-2*[*deduction*]
ord-eq-Eequiv-2[*deduction*] *identity-ν-def* **by** *metis*
}
ultimately show $[(x^P = y^P) \equiv (x^P =_E y^P) \ in \ v]$
using *intro-elim-4-a CP* **by** *blast*
qed

lemma *ord-eq-E*[*PLM*]:
$[((\!| O!,x^P |\!) \ \& \ (\!| O!,y^P |\!)) \to ((\forall \ F \ . \ (\!| F,x^P |\!) \equiv (\!| F,y^P |\!)) \to x^P =_E y^P) \ in \ v]$
proof (*rule CP*; *rule CP*)
assume *ord-xy*: $[(\!| O!,x^P |\!) \ \& \ (\!| O!,y^P |\!) \ in \ v]$
assume $[\forall \ F \ . \ (\!| F,x^P |\!) \equiv (\!| F,y^P |\!) \ in \ v]$
hence $[(\!| \boldsymbol{\lambda} \ z \ . \ z^P =_E x^P, x^P |\!) \equiv (\!| \boldsymbol{\lambda} \ z \ . \ z^P =_E x^P, y^P |\!) \ in \ v]$
by (*rule* $\forall E$)
moreover have $[(\!| \boldsymbol{\lambda} \ z \ . \ z^P =_E x^P, x^P |\!) \ in \ v]$
apply (*rule beta-C-meta-1*[*equiv-rl*])
unfolding *identity$_E$-infix-def*
apply *show-proper*
using *ord-eq-Eequiv-1*[*deduction*] *ord-xy*[*conj1*]
unfolding *identity$_E$-infix-def* **by** *simp*
ultimately have $[(\!| \boldsymbol{\lambda} \ z \ . \ z^P =_E x^P, y^P |\!) \ in \ v]$
using $\equiv E$ **by** *blast*
hence $[y^P =_E x^P \ in \ v]$
unfolding *identity$_E$-infix-def*
apply (*safe intro*!:
beta-C-meta-1[**where** $\varphi = \lambda \ z \ . \ (\!|$*basic-identity$_E$*$,z,x^P |\!)$, *equiv-lr*])
by *show-proper*
thus $[x^P =_E y^P \ in \ v]$
by (*rule ord-eq-Eequiv-2*[*deduction*])
qed

lemma *ord-eq-E2*[*PLM*]:
$[((\!| O!,x^P |\!) \ \& \ (\!| O!,y^P |\!)) \to$
$((x^P \neq y^P) \equiv (\boldsymbol{\lambda} z \ . \ z^P =_E x^P) \neq (\boldsymbol{\lambda} z \ . \ z^P =_E y^P)) \ in \ v]$
proof (*rule CP*; *rule* $\equiv I$; *rule CP*)
assume *ord-xy*: $[(\!| O!,x^P |\!) \ \& \ (\!| O!,y^P |\!) \ in \ v]$
assume $[x^P \neq y^P \ in \ v]$
hence $[\neg(x^P =_E y^P) \ in \ v]$
using *eq-E-simple-2 modus-tollens-1* **by** *fast*
moreover {
assume $[(\boldsymbol{\lambda} z \ . \ z^P =_E x^P) = (\boldsymbol{\lambda} z \ . \ z^P =_E y^P) \ in \ v]$
moreover have $[(\!| \boldsymbol{\lambda} z \ . \ z^P =_E x^P, x^P |\!) \ in \ v]$
apply (*rule beta-C-meta-1*[*equiv-rl*])
unfolding *identity$_E$-infix-def*
apply *show-proper*
using *ord-eq-Eequiv-1*[*deduction*] *ord-xy*[*conj1*]
unfolding *identity$_E$-infix-def* **by** *presburger*
ultimately have $[(\!| \boldsymbol{\lambda} z \ . \ z^P =_E y^P, x^P |\!) \ in \ v]$
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
hence $[x^P =_E y^P \ in \ v]$
unfolding *identity$_E$-infix-def*
apply (*safe intro*!:
beta-C-meta-1[**where** $\varphi = \lambda \ z \ . \ (\!|$*basic-identity$_E$*$,z,y^P |\!)$, *equiv-lr*])
by *show-proper*
}
ultimately show $[(\boldsymbol{\lambda} z \ . \ z^P =_E x^P) \neq (\boldsymbol{\lambda} z \ . \ z^P =_E y^P) \ in \ v]$
using *modus-tollens-1 CP* **by** *blast*
next
assume *ord-xy*: $[(\!| O!,x^P |\!) \ \& \ (\!| O!,y^P |\!) \ in \ v]$
assume $[(\boldsymbol{\lambda} z \ . \ z^P =_E x^P) \neq (\boldsymbol{\lambda} z \ . \ z^P =_E y^P) \ in \ v]$

moreover {
assume $[x^P = y^P \ in \ v]$
hence $[(\boldsymbol{\lambda} z \ . \ z^P =_E x^P) = (\boldsymbol{\lambda} z \ . \ z^P =_E y^P) \ in \ v]$
using *id-eq-1 l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *fast*
}
ultimately show $[x^P \neq y^P \ in \ v]$
using *modus-tollens-1 CP* **by** *blast*
qed

lemma *ab-obey-1*[*PLM*]:
$[((\!|A!,x^P|\!) \ \& \ (\!|A!,y^P|\!)) \rightarrow ((\forall \ F \ . \ \{\!|x^P, F|\!\} \equiv \{\!|y^P, F|\!\}) \rightarrow x^P = y^P) \ in \ v]$
proof(*rule CP*; *rule CP*)
assume *abs-xy*: $[(\!|A!,x^P|\!) \ \& \ (\!|A!,y^P|\!) \ in \ v]$
assume *enc-equiv*: $[\forall \ F \ . \ \{\!|x^P, F|\!\} \equiv \{\!|y^P, F|\!\} \ in \ v]$
{
fix P
have $[\{\!|x^P, P|\!\} \equiv \{\!|y^P, P|\!\} \ in \ v]$
using *enc-equiv* **by** (*rule* $\forall E$)
hence $[\Box(\{\!|x^P, P|\!\} \equiv \{\!|y^P, P|\!\}) \ in \ v]$
using *en-eq-2 intro-elim-6-e intro-elim-6-f*
en-eq-5[*equiv-rl*] **by** *meson*
}
hence $[\Box(\forall \ F \ . \ \{\!|x^P, F|\!\} \equiv \{\!|y^P, F|\!\}) \ in \ v]$
using *BF*[*deduction*] $\forall I$ **by** *fast*
thus $[x^P = y^P \ in \ v]$
unfolding *identity-defs*
using $\vee I(2)$ *abs-xy* **&***I* **by** *presburger*
qed

lemma *ab-obey-2*[*PLM*]:
$[((\!|A!,x^P|\!) \ \& \ (\!|A!,y^P|\!)) \rightarrow ((\exists \ F \ . \ \{\!|x^P, F|\!\} \ \& \ \neg\{\!|y^P, F|\!\}) \rightarrow x^P \neq y^P) \ in \ v]$
proof(*rule CP*; *rule CP*)
assume *abs-xy*: $[(\!|A!,x^P|\!) \ \& \ (\!|A!,y^P|\!) \ in \ v]$
assume $[\exists \ F \ . \ \{\!|x^P, F|\!\} \ \& \ \neg\{\!|y^P, F|\!\} \ in \ v]$
then obtain P **where** *P-prop*:
$[\{\!|x^P, P|\!\} \ \& \ \neg\{\!|y^P, P|\!\} \ in \ v]$
by (*rule* $\exists E$)
{
assume $[x^P = y^P \ in \ v]$
hence $[\{\!|x^P, P|\!\} \equiv \{\!|y^P, P|\!\} \ in \ v]$
using *l-identity*[*axiom-instance*, *deduction*, *deduction*]
oth-class-taut-4-a **by** *fast*
hence $[\{\!|y^P, P|\!\} \ in \ v]$
using *P-prop*[*conj1*] **by** (*rule* $\equiv E$)
}
thus $[x^P \neq y^P \ in \ v]$
using *P-prop*[*conj2*] *modus-tollens-1 CP* **by** *blast*
qed

lemma *ordnecfail*[*PLM*]:
$[(\!|O!,x^P|\!) \rightarrow \Box(\neg(\exists \ F \ . \ \{\!|x^P, F|\!\})) \ in \ v]$
proof (*rule CP*)
assume $[(\!|O!,x^P|\!) \ in \ v]$
hence $[\Box(\!|O!,x^P|\!) \ in \ v]$
using *oa-facts-1*[*deduction*] **by** *simp*
moreover hence $[\Box((\!|O!,x^P|\!) \rightarrow (\neg(\exists \ F \ . \ \{\!|x^P, F|\!\}))) \ in \ v]$
using *nocoder*[*axiom-necessitation*, *axiom-instance*] **by** *simp*
ultimately show $[\Box(\neg(\exists \ F \ . \ \{\!|x^P, F|\!\})) \ in \ v]$
using *qml-1*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
qed

lemma *o-objects-exist-1*[*PLM*]:

```
  [◇(∃ x . (|E!,x^P|)) in v]
  proof -
    have [◇(∃ x . (|E!,x^P|) & ◇(¬(|E!,x^P|))) in v]
      using qml-4[axiom-instance, conj1] .
    hence [◇((∃ x . (|E!,x^P|)) & (∃ x . ◇(¬(|E!,x^P|)))) in v]
      using sign-S5-thm-3[deduction] by fast
    hence [◇(∃ x . (|E!,x^P|)) & ◇(∃ x . ◇(¬(|E!,x^P|))) in v]
      using KBasic2-8[deduction] by blast
    thus ?thesis using &E by blast
  qed

lemma o-objects-exist-2[PLM]:
  [□(∃ x . (|O!,x^P|)) in v]
  apply (rule RN) unfolding Ordinary-def
  apply (PLM-subst-method λ x . ◇(|E!,x^P|) λ x . (|λy. ◇(|E!,y^P|), x^P|))
   apply (safe intro!: beta-C-meta-1[equiv-sym])
   apply show-proper
  using o-objects-exist-1 BF◇[deduction] by blast

lemma o-objects-exist-3[PLM]:
  [□(¬(∀ x . (|A!,x^P|))) in v]
  apply (PLM-subst-method (∃x. ¬(|A!,x^P|)) ¬(∀x. (|A!,x^P|)))
   using cqt-further-2[equiv-sym] apply fast
  apply (PLM-subst-method λ x . (|O!,x^P|) λ x . ¬(|A!,x^P|))
  using oa-contingent-2 o-objects-exist-2 by auto

lemma a-objects-exist-1[PLM]:
  [□(∃ x . (|A!,x^P|)) in v]
  proof -
    {
      fix v
      have [∃ x . (|A!,x^P|) & (∀ F . {|x^P, F|} ≡ (F = F)) in v]
        using A-objects[axiom-instance] by simp
      hence [∃ x . (|A!,x^P|) in v]
        using cqt-further-5[deduction,conj1] by fast
    }
    thus ?thesis by (rule RN)
  qed

lemma a-objects-exist-2[PLM]:
  [□(¬(∀ x . (|O!,x^P|))) in v]
  apply (PLM-subst-method (∃x. ¬(|O!,x^P|)) ¬(∀x. (|O!,x^P|)))
   using cqt-further-2[equiv-sym] apply fast
  apply (PLM-subst-method λ x . (|A!,x^P|) λ x . ¬(|O!,x^P|))
   using oa-contingent-3 a-objects-exist-1 by auto

lemma a-objects-exist-3[PLM]:
  [□(¬(∀ x . (|E!,x^P|))) in v]
  proof -
    {
      fix v
      have [∃ x . (|A!,x^P|) & (∀ F . {|x^P, F|} ≡ (F = F)) in v]
        using A-objects[axiom-instance] by simp
      hence [∃ x . (|A!,x^P|) in v]
        using cqt-further-5[deduction,conj1] by fast
      then obtain a where
        [(|A!,a^P|) in v]
        by (rule ∃E)
      hence [¬(◇(|E!,a^P|)) in v]
        unfolding Abstract-def
        apply (safe intro!: beta-C-meta-1[equiv-lr])
        by show-proper
      hence [(¬(|E!,a^P|)) in v]
```

    using *KBasic2-4*[*equiv-rl*] *qml-2*[*axiom-instance*,*deduction*]
    by *simp*
  hence $[\neg(\forall\ x\ .\ (|E!,x^P|))\ in\ v]$
    using $\exists I$ *cqt-further-2*[*equiv-rl*]
    by *fast*
 }
 thus *?thesis*
  by (*rule RN*)
qed

lemma *encoders-are-abstract*[*PLM*]:
 $[(\exists\ F\ .\ \{|x^P,\ F|\}) \rightarrow (|A!,x^P|)\ in\ v]$
 using *nocoder*[*axiom-instance*] *contraposition-2*
   oa-contingent-2[*THEN oth-class-taut-5-d*[*equiv-lr*], *equiv-lr*]
   useful-tautologies-1[*deduction*]
   vdash-properties-10 CP **by** *metis*

lemma *A-objects-unique*[*PLM*]:
 $[\exists !\ x\ .\ (|A!,x^P|)\ \&\ (\forall\ F\ .\ \{|x^P,\ F|\} \equiv \varphi\ F)\ in\ v]$
 proof −
  have $[\exists\ x\ .\ (|A!,x^P|)\ \&\ (\forall\ F\ .\ \{|x^P,\ F|\} \equiv \varphi\ F)\ in\ v]$
   using *A-objects*[*axiom-instance*] **by** *simp*
  then obtain a **where** *a-prop*:
   $[(|A!,a^P|)\ \&\ (\forall\ F\ .\ \{|a^P,\ F|\} \equiv \varphi\ F)\ in\ v]$ **by** (*rule* $\exists E$)
  moreover have $[\forall\ y\ .\ (|A!,y^P|)\ \&\ (\forall\ F\ .\ \{|y^P,\ F|\} \equiv \varphi\ F) \rightarrow (y = a)\ in\ v]$
   proof (*rule* $\forall I$; *rule CP*)
    fix b
    assume *b-prop*: $[(|A!,b^P|)\ \&\ (\forall\ F\ .\ \{|b^P,\ F|\} \equiv \varphi\ F)\ in\ v]$
    {
     fix P
     have $[\{|b^P,P|\} \equiv \{|a^P,\ P|\}\ in\ v]$
      using *a-prop*[*conj2*] *b-prop*[*conj2*] $\equiv I$ $\equiv E(1)$ $\equiv E(2)$
        CP vdash-properties-10 $\forall E$ **by** *metis*
    }
    hence $[\forall\ F\ .\ \{|b^P,F|\} \equiv \{|a^P,\ F|\}\ in\ v]$
     using $\forall I$ **by** *fast*
    thus $[b = a\ in\ v]$
     unfolding *identity-*ν*-def*
     using *ab-obey-1*[*deduction*, *deduction*]
       a-prop[*conj1*] *b-prop*[*conj1*] **&***I* **by** *blast*
   qed
  ultimately show *?thesis*
   unfolding *exists-unique-def*
   using **&***I* $\exists I$ **by** *fast*
 qed

lemma *obj-oth-1*[*PLM*]:
 $[\exists !\ x\ .\ (|A!,x^P|)\ \&\ (\forall\ F\ .\ \{|x^P,\ F|\} \equiv (|F,\ y^P|))\ in\ v]$
 using *A-objects-unique* **.**

lemma *obj-oth-2*[*PLM*]:
 $[\exists !\ x\ .\ (|A!,x^P|)\ \&\ (\forall\ F\ .\ \{|x^P,\ F|\} \equiv ((|F,\ y^P|)\ \&\ (|F,\ z^P|)))\ in\ v]$
 using *A-objects-unique* **.**

lemma *obj-oth-3*[*PLM*]:
 $[\exists !\ x\ .\ (|A!,x^P|)\ \&\ (\forall\ F\ .\ \{|x^P,\ F|\} \equiv ((|F,\ y^P|) \vee (|F,\ z^P|)))\ in\ v]$
 using *A-objects-unique* **.**

lemma *obj-oth-4*[*PLM*]:
 $[\exists !\ x\ .\ (|A!,x^P|)\ \&\ (\forall\ F\ .\ \{|x^P,\ F|\} \equiv (\Box(|F,\ y^P|)))\ in\ v]$
 using *A-objects-unique* **.**

lemma *obj-oth-5*[*PLM*]:

$[\exists !\ x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\}\equiv (F = G))\ \textit{in}\ v]$
using *A-objects-unique* **.**

lemma *obj-oth-6*[*PLM*]:
$[\exists !\ x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\}\equiv \Box(\forall\ y\ .\ (\!|G,\ y^P|\!)\rightarrow (\!|F,\ y^P|\!)))\ \textit{in}\ v]$
using *A-objects-unique* **.**

lemma *A-Exists-1*[*PLM*]:
$[\boldsymbol{\mathcal{A}}(\exists !\ x :: ('a :: \textit{id-act})\ .\ \varphi\ x) \equiv (\exists !\ x\ .\ \boldsymbol{\mathcal{A}}(\varphi\ x))\ \textit{in}\ v]$
unfolding *exists-unique-def*
proof (*rule* $\equiv I$; *rule CP*)
assume $[\boldsymbol{\mathcal{A}}(\exists\,\alpha.\ \varphi\ \alpha\ \&\ (\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha))\ \textit{in}\ v]$
hence $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}(\varphi\ \alpha\ \&\ (\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha))\ \textit{in}\ v]$
using *Act-Basic-11*[*equiv-lr*] **by** *blast*
then obtain α **where**
$[\boldsymbol{\mathcal{A}}(\varphi\ \alpha\ \&\ (\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha))\ \textit{in}\ v]$
by (*rule* $\exists\,E$)
hence *1*: $[\boldsymbol{\mathcal{A}}(\varphi\ \alpha)\ \&\ \boldsymbol{\mathcal{A}}(\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
using *Act-Basic-2*[*equiv-lr*] **by** *blast*
find-theorems $\boldsymbol{\mathcal{A}}(?p = ?q)$
have *2*: $[\forall\,\beta.\ \boldsymbol{\mathcal{A}}(\varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
using *1*[*conj2*] *logic-actual-nec-3*[*axiom-instance*, *equiv-lr*] **by** *blast*
{
fix β
have $[\boldsymbol{\mathcal{A}}(\varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
using *2* **by** (*rule* $\forall\,E$)
hence $[\boldsymbol{\mathcal{A}}(\varphi\ \beta) \rightarrow (\beta = \alpha)\ \textit{in}\ v]$
using *logic-actual-nec-2*[*axiom-instance*, *equiv-lr*, *deduction*]
id-act-3[*equiv-rl*] *CP* **by** *blast*
}
hence $[\forall\ \beta\ .\ \boldsymbol{\mathcal{A}}(\varphi\ \beta) \rightarrow (\beta = \alpha)\ \textit{in}\ v]$
by (*rule* $\forall\,I$)
thus $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}\varphi\ \alpha\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
using *1*[*conj1*] **&***I* $\exists\,I$ **by** *fast*
next
assume $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}\varphi\ \alpha\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
then obtain α **where** *1*:
$[\boldsymbol{\mathcal{A}}\varphi\ \alpha\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
by (*rule* $\exists\,E$)
{
fix β
have $[\boldsymbol{\mathcal{A}}(\varphi\ \beta) \rightarrow \beta = \alpha\ \textit{in}\ v]$
using *1*[*conj2*] **by** (*rule* $\forall\,E$)
hence $[\boldsymbol{\mathcal{A}}(\varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
using *logic-actual-nec-2*[*axiom-instance*, *equiv-rl*] *id-act-3*[*equiv-lr*]
vdash-properties-10 CP **by** *blast*
}
hence $[\forall\ \beta\ .\ \boldsymbol{\mathcal{A}}(\varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
by (*rule* $\forall\,I$)
hence $[\boldsymbol{\mathcal{A}}(\forall\ \beta\ .\ \varphi\ \beta \rightarrow \beta = \alpha)\ \textit{in}\ v]$
using *logic-actual-nec-3*[*axiom-instance*, *equiv-rl*] **by** *fast*
hence $[\boldsymbol{\mathcal{A}}(\varphi\ \alpha\ \&\ (\forall\ \beta\ .\ \varphi\ \beta \rightarrow \beta = \alpha))\ \textit{in}\ v]$
using *1*[*conj1*] *Act-Basic-2*[*equiv-rl*] **&***I* **by** *blast*
hence $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}(\varphi\ \alpha\ \&\ (\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha))\ \textit{in}\ v]$
using $\exists\,I$ **by** *fast*
thus $[\boldsymbol{\mathcal{A}}(\exists\,\alpha.\ \varphi\ \alpha\ \&\ (\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha))\ \textit{in}\ v]$
using *Act-Basic-11*[*equiv-rl*] **by** *fast*
qed

lemma *A-Exists-2*[*PLM*]:
$[(\exists\ y\ .\ y^P = (\iota x\ .\ \varphi\ x)) \equiv \boldsymbol{\mathcal{A}}(\exists !x\ .\ \varphi\ x)\ \textit{in}\ v]$
using *actual-desc-1 A-Exists-1*[*equiv-sym*]
intro-elim-6-e **by** *blast*

lemma *A-descriptions*[*PLM*]:
$[\exists\ y\ .\ y^P = (\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F))\ in\ v]$
using *A-objects-unique*[*THEN RN*, *THEN nec-imp-act*[*deduction*]]
A-Exists-2[*equiv-rl*] **by** *auto*

lemma *thm-can-terms2*[*PLM*]:
$[(y^P = (\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F)))$
$\to ((\!|A!,y^P|\!)\ \&\ (\forall\ F\ .\ \{\!|y^P,F|\!\} \equiv \varphi\ F))\ in\ dw]$
using *y-in-2* **by** *auto*

lemma *can-ab2*[*PLM*]:
$[(y^P = (\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F))) \to (\!|A!,y^P|\!)\ in\ v]$
proof (*rule CP*)
assume $[y^P = (\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F))\ in\ v]$
hence $[\boldsymbol{\mathcal{A}}(\!|A!,y^P|\!)\ \&\ \boldsymbol{\mathcal{A}}(\forall\ F\ .\ \{\!|y^P,F|\!\} \equiv \varphi\ F)\ in\ v]$
using *nec-hintikka-scheme*[*equiv-lr*, *conj1*]
Act-Basic-2[*equiv-lr*] **by** *blast*
thus $[(\!|A!,y^P|\!)\ in\ v]$
using *oa-facts-8*[*equiv-rl*] $\&E$ **by** *blast*
qed

lemma *desc-encode*[*PLM*]:
$[\{\!|\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F),\ G|\!\} \equiv \varphi\ G\ in\ dw]$
proof −
obtain a **where**
$[a^P = (\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F))\ in\ dw]$
using *A-descriptions* **by** (*rule* $\exists E$)
moreover hence $[\{\!|a^P,\ G|\!\} \equiv \varphi\ G\ in\ dw]$
using *hintikka*[*equiv-lr*, *conj1*] $\&E\ \forall E$ **by** *fast*
ultimately show *?thesis*
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
qed

lemma *desc-nec-encode*[*PLM*]:
$[\{\!|\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F),\ G|\!\} \equiv \boldsymbol{\mathcal{A}}(\varphi\ G)\ in\ v]$
proof −
obtain a **where**
$[a^P = (\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F))\ in\ v]$
using *A-descriptions* **by** (*rule* $\exists E$)
moreover {
hence $[\boldsymbol{\mathcal{A}}((\!|A!,a^P|\!)\ \&\ (\forall\ F\ .\ \{\!|a^P,F|\!\} \equiv \varphi\ F))\ in\ v]$
using *nec-hintikka-scheme*[*equiv-lr*, *conj1*] **by** *fast*
hence $[\boldsymbol{\mathcal{A}}(\forall\ F\ .\ \{\!|a^P,F|\!\} \equiv \varphi\ F)\ in\ v]$
using *Act-Basic-2*[*equiv-lr*,*conj2*] **by** *blast*
hence $[\forall\ F\ .\ \boldsymbol{\mathcal{A}}(\ \{\!|a^P,F|\!\} \equiv \varphi\ F)\ in\ v]$
using *logic-actual-nec-3*[*axiom-instance*, *equiv-lr*] **by** *blast*
hence $[\boldsymbol{\mathcal{A}}(\{\!|a^P,\ G|\!\} \equiv \varphi\ G)\ in\ v]$
using $\forall E$ **by** *fast*
hence $[\boldsymbol{\mathcal{A}}\{\!|a^P,\ G|\!\} \equiv \boldsymbol{\mathcal{A}}(\varphi\ G)\ in\ v]$
using *Act-Basic-5*[*equiv-lr*] **by** *fast*
hence $[\{\!|a^P,\ G|\!\} \equiv \boldsymbol{\mathcal{A}}(\varphi\ G)\ in\ v]$
using *en-eq-10*[*equiv-sym*] *intro-elim-6-e* **by** *blast*
}
ultimately show *?thesis*
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
qed

notepad
begin
fix v
let $?x = \iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv (\exists\ q\ .\ q\ \&\ F = (\boldsymbol{\lambda}\ y\ .\ q)))$
have $[\Box(\exists\ p\ .\ ContingentlyTrue\ p)\ in\ v]$

using *cont-tf-thm-3 RN* **by** *auto*
hence $[\boldsymbol{\mathcal{A}}(\exists\ p\ .\ \mathit{ContingentlyTrue}\ p)\ \mathit{in}\ v]$
using *nec-imp-act*[*deduction*] **by** *simp*
hence $[\exists\ p\ .\ \boldsymbol{\mathcal{A}}(\mathit{ContingentlyTrue}\ p)\ \mathit{in}\ v]$
using *Act-Basic-11*[*equiv-lr*] **by** *auto*
then obtain p_1 **where**
$[\boldsymbol{\mathcal{A}}(\mathit{ContingentlyTrue}\ p_1)\ \mathit{in}\ v]$
by (*rule* $\exists E$)
hence $[\boldsymbol{\mathcal{A}} p_1\ \mathit{in}\ v]$
unfolding *ContingentlyTrue-def*
using *Act-Basic-2*[*equiv-lr*] $\&E$ **by** *fast*
hence $[\boldsymbol{\mathcal{A}} p_1\ \&\ \boldsymbol{\mathcal{A}}((\boldsymbol{\lambda}\ y\ .\ p_1) = (\boldsymbol{\lambda}\ y\ .\ p_1))\ \mathit{in}\ v]$
using $\&I$ *id-eq-1*[*THEN RN*, *THEN nec-imp-act*[*deduction*]] **by** *fast*
hence $[\boldsymbol{\mathcal{A}}(p_1\ \&\ (\boldsymbol{\lambda}\ y\ .\ p_1) = (\boldsymbol{\lambda}\ y\ .\ p_1))\ \mathit{in}\ v]$
using *Act-Basic-2*[*equiv-rl*] **by** *fast*
hence $[\exists\ q\ .\ \boldsymbol{\mathcal{A}}(\ q\ \&\ (\boldsymbol{\lambda}\ y\ .\ p_1) = (\boldsymbol{\lambda}\ y\ .\ q))\ \mathit{in}\ v]$
using $\exists I$ **by** *fast*
hence $[\boldsymbol{\mathcal{A}}(\exists\ q\ .\ q\ \&\ (\boldsymbol{\lambda}\ y\ .\ p_1) = (\boldsymbol{\lambda}\ y\ .\ q))\ \mathit{in}\ v]$
using *Act-Basic-11*[*equiv-rl*] **by** *fast*
moreover have $[\{\!|\ ?x,\ \boldsymbol{\lambda}\ y\ .\ p_1|\!\} \equiv \boldsymbol{\mathcal{A}}(\exists\ q\ .\ q\ \&\ (\boldsymbol{\lambda}\ y\ .\ p_1) = (\boldsymbol{\lambda}\ y\ .\ q))\ \mathit{in}\ v]$
using *desc-nec-encode* **by** *fast*
ultimately have $[\{\!|\ ?x,\ \boldsymbol{\lambda}\ y\ .\ p_1|\!\}\ \mathit{in}\ v]$
using $\equiv E$ **by** *blast*
end

lemma *Box-desc-encode-1*[*PLM*]:
$[\Box(\varphi\ G) \rightarrow \{\!|(\boldsymbol{\iota} x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\}\ \mathit{in}\ v]$
proof (*rule CP*)
assume $[\Box(\varphi\ G)\ \mathit{in}\ v]$
hence $[\boldsymbol{\mathcal{A}}(\varphi\ G)\ \mathit{in}\ v]$
using *nec-imp-act*[*deduction*] **by** *auto*
thus $[\{\!|\boldsymbol{\iota} x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F),\ G|\!\}\ \mathit{in}\ v]$
using *desc-nec-encode*[*equiv-rl*] **by** *simp*
qed

lemma *Box-desc-encode-2*[*PLM*]:
$[\Box(\varphi\ G) \rightarrow \Box(\{\!|(\boldsymbol{\iota} x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\} \equiv \varphi\ G)\ \mathit{in}\ v]$
proof (*rule CP*)
assume *a*: $[\Box(\varphi\ G)\ \mathit{in}\ v]$
hence $[\Box(\{\!|(\boldsymbol{\iota} x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\} \rightarrow \varphi\ G)\ \mathit{in}\ v]$
using *KBasic-1*[*deduction*] **by** *simp*
moreover **{**
have $[\{\!|(\boldsymbol{\iota} x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\}\ \mathit{in}\ v]$
using *a Box-desc-encode-1*[*deduction*] **by** *auto*
hence $[\Box\{\!|(\boldsymbol{\iota} x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\}\ \mathit{in}\ v]$
using *encoding*[*axiom-instance*,*deduction*] **by** *blast*
hence $[\Box(\varphi\ G \rightarrow\ \{\!|(\boldsymbol{\iota} x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\})\ \mathit{in}\ v]$
using *KBasic-1*[*deduction*] **by** *simp*
}
ultimately show $[\Box(\{\!|(\boldsymbol{\iota} x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\}$
$\equiv \varphi\ G)\ \mathit{in}\ v]$
using $\&I$ *KBasic-4*[*equiv-rl*] **by** *blast*
qed

lemma *box-phi-a-1*[*PLM*]:
assumes $[\Box(\forall\ F\ .\ \varphi\ F \rightarrow \Box(\varphi\ F))\ \mathit{in}\ v]$
shows $[((\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)) \rightarrow \Box((\!|A!,x^P|\!)$
$\&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F))\ \mathit{in}\ v]$
proof (*rule CP*)
assume *a*: $[((\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F))\ \mathit{in}\ v]$
have $[\Box(\!|A!,x^P|\!)\ \mathit{in}\ v]$
using *oa-facts-2*[*deduction*] *a*[*conj1*] **by** *auto*
moreover have $[\Box(\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)\ \mathit{in}\ v]$

proof (*rule BF*[*deduction*]; *rule* $\forall I$)
fix F
have ϑ: $[\Box(\varphi\ F \rightarrow \Box(\varphi\ F))\ in\ v]$
using *assms*[*THEN CBF*[*deduction*]] **by** (*rule* $\forall E$)
moreover have $[\Box(\{\!\!\{x^P,\ F\}\!\!\} \rightarrow \Box\{\!\!\{x^P,\ F\}\!\!\})\ in\ v]$
using *encoding*[*axiom-necessitation*, *axiom-instance*] **by** *simp*
moreover have $[\Box\{\!\!\{x^P,\ F\}\!\!\} \equiv \Box(\varphi\ F)\ in\ v]$
proof (*rule* $\equiv I$; *rule CP*)
assume $[\Box\{\!\!\{x^P,\ F\}\!\!\}\ in\ v]$
hence $[\{\!\!\{x^P,\ F\}\!\!\}\ in\ v]$
using *qml-2*[*axiom-instance*, *deduction*] **by** *blast*
hence $[\varphi\ F\ in\ v]$
using *a*[*conj2*] $\forall E$[**where** $'a{=}\Pi_1$] $\equiv E$ **by** *blast*
thus $[\Box(\varphi\ F)\ in\ v]$
using ϑ[*THEN qml-2*[*axiom-instance*, *deduction*], *deduction*] **by** *simp*
next
assume $[\Box(\varphi\ F)\ in\ v]$
hence $[\varphi\ F\ in\ v]$
using *qml-2*[*axiom-instance*, *deduction*] **by** *blast*
hence $[\{\!\!\{x^P,\ F\}\!\!\}\ in\ v]$
using *a*[*conj2*] $\forall E$[**where** $'a{=}\Pi_1$] $\equiv E$ **by** *blast*
thus $[\Box\{\!\!\{x^P,\ F\}\!\!\}\ in\ v]$
using *encoding*[*axiom-instance*, *deduction*] **by** *simp*
qed
ultimately show $[\Box(\{\!\!\{x^P,F\}\!\!\} \equiv \varphi\ F)\ in\ v]$
using *sc-eq-box-box-3*[*deduction*, *deduction*] $\&I$ **by** *blast*
qed
ultimately show $[\Box((\!|A!,x^P|\!)\ \&\ (\forall F.\ \{\!\!\{x^P,F\}\!\!\} \equiv \varphi\ F))\ in\ v]$
using $\&I$ *KBasic-3*[*equiv-rl*] **by** *blast*
qed

lemma *box-phi-a-2*[*PLM*]:
assumes $[\Box(\forall\ F\ .\ \varphi\ F \rightarrow \Box(\varphi\ F))\ in\ v]$
shows $[y^P = (\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F.\ \{\!\!\{x^P,\ F\}\!\!\} \equiv \varphi\ F))$
$\rightarrow ((\!|A!,y^P|\!)\ \&\ (\forall\ F\ .\ \{\!\!\{y^P,\ F\}\!\!\} \equiv \varphi\ F))\ in\ v]$
proof −
let $?\psi = \lambda\ x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!\!\{x^P,\ F\}\!\!\} \equiv \varphi\ F)$
have $[\forall\ x\ .\ ?\psi\ x \rightarrow \Box(?\psi\ x)\ in\ v]$
using *box-phi-a-1*[*OF assms*] $\forall I$ **by** *fast*
hence $[(\exists!\ x\ .\ ?\psi\ x) \rightarrow (\forall\ y\ .\ y^P = (\iota x\ .\ ?\psi\ x) \rightarrow ?\psi\ y)\ in\ v]$
using *unique-box-desc*[*deduction*] **by** *fast*
hence $[(\forall\ y\ .\ y^P = (\iota x\ .\ ?\psi\ x) \rightarrow ?\psi\ y)\ in\ v]$
using *A-objects-unique modus-ponens* **by** *blast*
thus *?thesis* **by** (*rule* $\forall E$)
qed

lemma *box-phi-a-3*[*PLM*]:
assumes $[\Box(\forall\ F\ .\ \varphi\ F \rightarrow \Box(\varphi\ F))\ in\ v]$
shows $[\{\!\!\{\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!\!\{x^P,\ F\}\!\!\} \equiv \varphi\ F),\ G\}\!\!\} \equiv \varphi\ G\ in\ v]$
proof −
obtain a **where**
$[a^P = (\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!\!\{x^P,\ F\}\!\!\} \equiv \varphi\ F))\ in\ v]$
using *A-descriptions* **by** (*rule* $\exists E$)
moreover {
hence $[(\forall\ F\ .\ \{\!\!\{a^P,\ F\}\!\!\} \equiv \varphi\ F)\ in\ v]$
using *box-phi-a-2*[*OF assms*, *deduction*, *conj2*] **by** *blast*
hence $[\{\!\!\{a^P,\ G\}\!\!\} \equiv \varphi\ G\ in\ v]$ **by** (*rule* $\forall E$)
}
ultimately show *?thesis*
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
qed

lemma *null-uni-uniq-1*[*PLM*]:

[$\exists$! x . *Null* (x^P) *in v*]
proof –
have [$\exists$ x . ⦇$A!$,x^P⦈ **&** ($\forall$ F . ⦃x^P, F⦄ $\equiv$ ($F \neq F$)) *in v*]
using *A-objects*[*axiom-instance*] **by** *simp*
then obtain a **where** *a-prop*:
[⦇$A!$,a^P⦈ **&** ($\forall$ F . ⦃a^P, F⦄ $\equiv$ ($F \neq F$)) *in v*]
by (*rule* $\exists E$)
have *1*: [⦇$A!$,a^P⦈ **&** ($\neg$($\exists$ F . ⦃a^P, F⦄)) *in v*]
using *a-prop*[*conj1*] **apply** (*rule* **&***I*)
proof –
{
assume [$\exists$ F . ⦃a^P, F⦄ *in v*]
then obtain P **where**
[⦃a^P, P⦄ *in v*] **by** (*rule* $\exists E$)
hence [$P \neq P$ *in v*]
using *a-prop*[*conj2*, *THEN* $\forall E$, *equiv-lr*] **by** *simp*
hence [$\neg$($\exists$ F . ⦃a^P, F⦄) *in v*]
using *id-eq-1 reductio-aa-1* **by** *fast*
}
thus [$\neg$($\exists$ F . ⦃a^P, F⦄) *in v*]
using *reductio-aa-1* **by** *blast*
qed
moreover have [$\forall$ y . (⦇$A!$,y^P⦈ **&** ($\neg$($\exists$ F . ⦃y^P, F⦄))) $\rightarrow$ $y = a$ *in v*]
proof (*rule* $\forall I$; *rule CP*)
fix y
assume *2*: [⦇$A!$,y^P⦈ **&** ($\neg$($\exists$ F . ⦃y^P, F⦄)) *in v*]
have [$\forall$ F . ⦃y^P, F⦄ $\equiv$ ⦃a^P, F⦄ *in v*]
using *cqt-further-12*[*deduction*] *1*[*conj2*] *2*[*conj2*] **&***I* **by** *blast*
thus [$y = a$ *in v*]
using *ab-obey-1*[*deduction*, *deduction*]
&*I*[*OF 2*[*conj1*] *1*[*conj1*]] *identity-ν-def* **by** *presburger*
qed
ultimately show *?thesis*
using **&***I* $\exists I$
unfolding *Null-def exists-unique-def* **by** *fast*
qed

lemma *null-uni-uniq-2*[*PLM*]:
[$\exists$! x . *Universal* (x^P) *in v*]
proof –
have [$\exists$ x . ⦇$A!$,x^P⦈ **&** ($\forall$ F . ⦃x^P, F⦄ $\equiv$ ($F = F$)) *in v*]
using *A-objects*[*axiom-instance*] **by** *simp*
then obtain a **where** *a-prop*:
[⦇$A!$,a^P⦈ **&** ($\forall$ F . ⦃a^P, F⦄ $\equiv$ ($F = F$)) *in v*]
by (*rule* $\exists E$)
have *1*: [⦇$A!$,a^P⦈ **&** ($\forall$ F . ⦃a^P, F⦄) *in v*]
using *a-prop*[*conj1*] **apply** (*rule* **&***I*)
using $\forall I$ *a-prop*[*conj2*, *THEN* $\forall E$, *equiv-rl*] *id-eq-1* **by** *fast*
moreover have [$\forall$ y . (⦇$A!$,y^P⦈ **&** ($\forall$ F . ⦃y^P, F⦄)) $\rightarrow$ $y = a$ *in v*]
proof (*rule* $\forall I$; *rule CP*)
fix y
assume *2*: [⦇$A!$,y^P⦈ **&** ($\forall$ F . ⦃y^P, F⦄) *in v*]
have [$\forall$ F . ⦃y^P, F⦄ $\equiv$ ⦃a^P, F⦄ *in v*]
using *cqt-further-11*[*deduction*] *1*[*conj2*] *2*[*conj2*] **&***I* **by** *blast*
thus [$y = a$ *in v*]
using *ab-obey-1*[*deduction*, *deduction*]
&*I*[*OF 2*[*conj1*] *1*[*conj1*]] *identity-ν-def*
by *presburger*
qed
ultimately show *?thesis*
using **&***I* $\exists I$
unfolding *Universal-def exists-unique-def* **by** *fast*
qed

```
lemma null-uni-uniq-3[PLM]:
  [∃ y . y^P = (ιx . Null (x^P)) in v]
  using null-uni-uniq-1[THEN RN, THEN nec-imp-act[deduction]]
        A-Exists-2[equiv-rl] by auto

lemma null-uni-uniq-4[PLM]:
  [∃ y . y^P = (ιx . Universal (x^P)) in v]
  using null-uni-uniq-2[THEN RN, THEN nec-imp-act[deduction]]
        A-Exists-2[equiv-rl] by auto

lemma null-uni-facts-1[PLM]:
  [Null (x^P) → □(Null (x^P)) in v]
  proof (rule CP)
    assume [Null (x^P) in v]
    hence 1: [(|A!,x^P|) & (¬(∃ F . {|x^P,F|})) in v]
      unfolding Null-def .
    have [□(|A!,x^P|) in v]
      using 1[conj1] oa-facts-2[deduction] by simp
    moreover have [□(¬(∃ F . {|x^P,F|})) in v]
      proof −
        {
          assume [¬□(¬(∃ F . {|x^P,F|})) in v]
          hence [◇(∃ F . {|x^P,F|}) in v]
            unfolding diamond-def .
          hence [∃ F . ◇{|x^P,F|} in v]
            using BF◇[deduction] by blast
          then obtain P where [◇{|x^P,P|} in v]
            by (rule ∃ E)
          hence [{|x^P, P|} in v]
            using en-eq-3[equiv-lr] by simp
          hence [∃  F . {|x^P, F|} in v]
            using ∃ I by fast
        }
        thus ?thesis
          using 1[conj2] modus-tollens-1 CP
                useful-tautologies-1[deduction] by metis
      qed
    ultimately show [□Null (x^P) in v]
      unfolding Null-def
      using &I KBasic-3[equiv-rl] by blast
  qed

lemma null-uni-facts-2[PLM]:
  [Universal (x^P) → □(Universal (x^P)) in v]
  proof (rule CP)
    assume [Universal (x^P) in v]
    hence 1: [(|A!,x^P|) & (∀ F . {|x^P,F|}) in v]
      unfolding Universal-def .
    have [□(|A!,x^P|) in v]
      using 1[conj1] oa-facts-2[deduction] by simp
    moreover have [□(∀ F . {|x^P,F|}) in v]
      proof (rule BF[deduction]; rule ∀ I)
        fix F
        have [{|x^P, F|} in v]
          using 1[conj2] by (rule ∀ E)
        thus [□{|x^P, F|} in v]
          using encoding[axiom-instance, deduction] by auto
      qed
    ultimately show [□Universal (x^P) in v]
      unfolding Universal-def
      using &I KBasic-3[equiv-rl] by blast
  qed
```

lemma *null-uni-facts-3*[*PLM*]:
[*Null* ($\mathbf{a}_\emptyset$) *in v*]
proof −
let $?\psi = \lambda\ x\ .\ Null\ x$
have $[((\exists!\ x\ .\ ?\psi\ (x^P)) \rightarrow (\forall\ y\ .\ y^P = (\iota x\ .\ ?\psi\ (x^P)) \rightarrow ?\psi\ (y^P)))\ in\ v]$
using *unique-box-desc*[*deduction*] *null-uni-facts-1*[*THEN* ∀ *I*] **by** *fast*
have *1*: $[(\forall\ y\ .\ y^P = (\iota x\ .\ ?\psi\ (x^P)) \rightarrow ?\psi\ (y^P))\ in\ v]$
using *unique-box-desc*[*deduction*, *deduction*] *null-uni-uniq-1*
null-uni-facts-1[*THEN* ∀ *I*] **by** *fast*
have $[\exists\ y\ .\ y^P = (\mathbf{a}_\emptyset)\ in\ v]$
unfolding *NullObject-def* **using** *null-uni-uniq-3* **.**
then obtain *y* **where** $[y^P = (\mathbf{a}_\emptyset)\ in\ v]$
by (*rule* ∃ *E*)
moreover hence $[?\psi\ (y^P)\ in\ v]$
using *1*[*THEN* ∀ *E*, *deduction*] **unfolding** *NullObject-def* **by** *simp*
ultimately show $[?\psi\ (\mathbf{a}_\emptyset)\ in\ v]$
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *blast*
qed

lemma *null-uni-facts-4*[*PLM*]:
[*Universal* ($\mathbf{a}_V$) *in v*]
proof −
let $?\psi = \lambda\ x\ .\ Universal\ x$
have $[((\exists!\ x\ .\ ?\psi\ (x^P)) \rightarrow (\forall\ y\ .\ y^P = (\iota x\ .\ ?\psi\ (x^P)) \rightarrow ?\psi\ (y^P)))\ in\ v]$
using *unique-box-desc*[*deduction*] *null-uni-facts-2*[*THEN* ∀ *I*] **by** *fast*
have *1*: $[(\forall\ y\ .\ y^P = (\iota x\ .\ ?\psi\ (x^P)) \rightarrow ?\psi\ (y^P))\ in\ v]$
using *unique-box-desc*[*deduction*, *deduction*] *null-uni-uniq-2*
null-uni-facts-2[*THEN* ∀ *I*] **by** *fast*
have $[\exists\ y\ .\ y^P = (\mathbf{a}_V)\ in\ v]$
unfolding *UniversalObject-def* **using** *null-uni-uniq-4* **.**
then obtain *y* **where** $[y^P = (\mathbf{a}_V)\ in\ v]$
by (*rule* ∃ *E*)
moreover hence $[?\psi\ (y^P)\ in\ v]$
using *1*[*THEN* ∀ *E*, *deduction*]
unfolding *UniversalObject-def* **by** *simp*
ultimately show $[?\psi\ (\mathbf{a}_V)\ in\ v]$
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *blast*
qed

lemma *aclassical-1*[*PLM*]:
$[\forall\ R\ .\ \exists\ x\ y\ .\ (|A!,x^P|)\ \&\ (|A!,y^P|)\ \&\ (x \neq y)$
$\&\ (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,x^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,y^P|))\ in\ v]$
proof (*rule* ∀ *I*)
fix *R*
obtain *a* **where** ϑ:
$[(|A!,a^P|)\ \&\ (\forall\ F\ .\ \{|a^P,\ F|\} \equiv (\exists\ y\ .\ (|A!,y^P|)$
$\&\ F = (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,y^P|))\ \&\ \neg\{|y^P,\ F|\}))\ in\ v]$
using *A-objects*[*axiom-instance*] **by** (*rule* ∃ *E*)
{
assume $[\neg\{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))|\}\ in\ v]$
hence $[\neg((|A!,a^P|)\ \&\ (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))$
$\&\ \neg\{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))|\})\ in\ v]$
using ϑ[*conj2*, *THEN* ∀ *E*, *THEN* *oth-class-taut-5-d*[*equiv-lr*], *equiv-lr*]
cqt-further-4[*equiv-lr*] ∀ *E* **by** *fast*
hence $[(|A!,a^P|)\ \&\ (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))$
$\rightarrow \{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))|\}\ in\ v]$
apply − **by** *PLM-solver*
hence $[\{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))|\}\ in\ v]$
using ϑ[*conj1*] *id-eq-1* **&***I* *vdash-properties-10* **by** *fast*
}
hence *1*: $[\{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))|\}\ in\ v]$
using *reductio-aa-1 CP if-p-then-p* **by** *blast*

then obtain b **where** ξ:
$[(|A!,b^P|)$ **&** $(\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,b^P|))$
& $\neg\{|b^P, (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))|\}$ *in* $v]$
using ϑ[*conj2*, *THEN* $\forall$ *E*, *equiv-lr*] $\exists$ *E* **by** *blast*
have $[a \neq b$ *in* $v]$
proof −
{
assume $[a = b$ *in* $v]$
hence $[\{|b^P, (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|))|\}$ *in* $v]$
using *1 l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
hence *?thesis*
using ξ[*conj2*] *reductio-aa-1* **by** *blast*
}
thus *?thesis* **using** *reductio-aa-1* **by** *blast*
qed
hence $[(|A!,a^P|)$ **&** $(|A!,b^P|)$ **&** $a \neq b$
& $(\boldsymbol{\lambda}\ z\ .\ (|R,z^P,a^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,z^P,b^P|))$ *in* $v]$
using ϑ[*conj1*] ξ[*conj1*, *conj1*] ξ[*conj1*, *conj2*] **&***I* **by** *presburger*
hence $[\exists\ y\ .\ (|A!,a^P|)$ **&** $(|A!,y^P|)$ **&** $a \neq y$
& $(\boldsymbol{\lambda} z.\ (|R,z^P,a^P|)) = (\boldsymbol{\lambda} z.\ (|R,z^P,y^P|))$ *in* $v]$
using $\exists$ *I* **by** *fast*
thus $[\exists\ x\ y\ .\ (|A!,x^P|)$ **&** $(|A!,y^P|)$ **&** $x \neq y$
& $(\boldsymbol{\lambda} z.\ (|R,z^P,x^P|)) = (\boldsymbol{\lambda} z.\ (|R,z^P,y^P|))$ *in* $v]$
using $\exists$ *I* **by** *fast*
qed

lemma *aclassical-2*[*PLM*]:
$[\forall\ R\ .\ \exists\ x\ y\ .\ (|A!,x^P|)$ **&** $(|A!,y^P|)$ **&** $(x \neq y)$
& $(\boldsymbol{\lambda}\ z\ .\ (|R,x^P,z^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,y^P,z^P|))$ *in* $v]$
proof (*rule* $\forall$ *I*)
fix R
obtain a **where** ϑ:
$[(|A!,a^P|)$ **&** $(\forall\ F\ .\ \{|a^P, F|\} \equiv (\exists\ y\ .\ (|A!,y^P|)$
& $F = (\boldsymbol{\lambda}\ z\ .\ (|R,y^P,z^P|))$ **&** $\neg\{|y^P, F|\}))$ *in* $v]$
using *A-objects*[*axiom-instance*] **by** (*rule* $\exists$ *E*)
{
assume $[\neg\{|a^P, (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))|\}$ *in* $v]$
hence $[\neg((|A!,a^P|)$ **&** $(\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))$
& $\neg\{|a^P, (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))|\})$ *in* $v]$
using ϑ[*conj2*, *THEN* $\forall$ *E*, *THEN oth-class-taut-5-d*[*equiv-lr*], *equiv-lr*]
cqt-further-4[*equiv-lr*] $\forall$ *E* **by** *fast*
hence $[(|A!,a^P|)$ **&** $(\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))$
$\rightarrow \{|a^P, (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))|\}$ *in* $v]$
apply − **by** *PLM-solver*
hence $[\{|a^P, (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))|\}$ *in* $v]$
using ϑ[*conj1*] *id-eq-1* **&***I* *vdash-properties-10* **by** *fast*
}
hence *1*: $[\{|a^P, (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))|\}$ *in* $v]$
using *reductio-aa-1 CP if-p-then-p* **by** *blast*
then obtain b **where** ξ:
$[(|A!,b^P|)$ **&** $(\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,b^P,z^P|))$
& $\neg\{|b^P, (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))|\}$ *in* $v]$
using ϑ[*conj2*, *THEN* $\forall$ *E*, *equiv-lr*] $\exists$ *E* **by** *blast*
have $[a \neq b$ *in* $v]$
proof −
{
assume $[a = b$ *in* $v]$
hence $[\{|b^P, (\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|))|\}$ *in* $v]$
using *1 l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
hence *?thesis* **using** ξ[*conj2*] *reductio-aa-1* **by** *blast*
}
thus *?thesis* **using** ξ[*conj2*] *reductio-aa-1* **by** *blast*
qed

```
    hence [⦇A!,a^P⦈ & ⦇A!,b^P⦈ & a ≠ b
          & (λ z . ⦇R,a^P,z^P⦈) = (λ z . ⦇R,b^P,z^P⦈) in v]
      using ϑ[conj1] ξ[conj1, conj1] ξ[conj1, conj2] &I by presburger
    hence [∃ y . ⦇A!,a^P⦈ & ⦇A!,y^P⦈ & a ≠ y
          & (λz. ⦇R,a^P,z^P⦈) = (λz. ⦇R,y^P,z^P⦈) in v]
      using ∃ I by fast
    thus [∃ x y . ⦇A!,x^P⦈ & ⦇A!,y^P⦈ & x ≠ y
          & (λz. ⦇R,x^P,z^P⦈) = (λz. ⦇R,y^P,z^P⦈) in v]
      using ∃ I by fast
  qed

lemma aclassical-3[PLM]:
  [∀ F . ∃ x y . ⦇A!,x^P⦈ & ⦇A!,y^P⦈ & (x ≠ y)
    & ((λ^0 ⦇F,x^P⦈) = (λ^0 ⦇F,y^P⦈)) in v]
  proof (rule ∀ I)
    fix R
    obtain a where ϑ:
      [⦇A!,a^P⦈ & (∀ F . ⦃a^P, F⦄ ≡ (∃ y . ⦇A!,y^P⦈
        & F = (λ z . ⦇R,y^P⦈) & ¬⦃y^P, F⦄)) in v]
      using A-objects[axiom-instance] by (rule ∃ E)
    {
      assume [¬⦃a^P, (λ z . ⦇R,a^P⦈)⦄ in v]
      hence [¬(⦇A!,a^P⦈ & (λ z . ⦇R,a^P⦈) = (λ z . ⦇R,a^P⦈)
            & ¬⦃a^P, (λ z . ⦇R,a^P⦈)⦄) in v]
        using ϑ[conj2, THEN ∀ E, THEN oth-class-taut-5-d[equiv-lr], equiv-lr]
              cqt-further-4[equiv-lr] ∀ E by fast
      hence [⦇A!,a^P⦈ & (λ z . ⦇R,a^P⦈) = (λ z . ⦇R,a^P⦈)
            → ⦃a^P, (λ z . ⦇R,a^P⦈)⦄ in v]
        apply − by PLM-solver
      hence [⦃a^P, (λ z . ⦇R,a^P⦈)⦄ in v]
        using ϑ[conj1] id-eq-1 &I vdash-properties-10 by fast
    }
    hence 1: [⦃a^P, (λ z . ⦇R,a^P⦈)⦄ in v]
      using reductio-aa-1 CP if-p-then-p by blast
    then obtain b where ξ:
      [⦇A!,b^P⦈ & (λ z . ⦇R,a^P⦈) = (λ z . ⦇R,b^P⦈)
        & ¬⦃b^P, (λ z . ⦇R,a^P⦈)⦄ in v]
      using ϑ[conj2, THEN ∀ E, equiv-lr] ∃ E by blast
    have [a ≠ b in v]
      proof −
        {
          assume [a = b in v]
          hence [⦃b^P, (λ z . ⦇R,a^P⦈)⦄ in v]
            using 1 l-identity[axiom-instance, deduction, deduction] by fast
          hence ?thesis
            using ξ[conj2] reductio-aa-1 by blast
        }
        thus ?thesis using reductio-aa-1 by blast
      qed
    moreover {
      have [⦇R,a^P⦈ = ⦇R,b^P⦈ in v]
        unfolding identity∘-def
        using ξ[conj1, conj2] by auto
      hence [(λ^0 ⦇R,a^P⦈) = (λ^0 ⦇R,b^P⦈) in v]
        using lambda-p-q-p-eq-q[equiv-rl] by simp
    }
    ultimately have [⦇A!,a^P⦈ & ⦇A!,b^P⦈ & a ≠ b
            & ((λ^0 ⦇R,a^P⦈) =(λ^0 ⦇R,b^P⦈)) in v]
      using ϑ[conj1] ξ[conj1, conj1] ξ[conj1, conj2] &I
      by presburger
    hence [∃ y . ⦇A!,a^P⦈ & ⦇A!,y^P⦈ & a ≠ y
          & (λ^0 ⦇R,a^P⦈) = (λ^0 ⦇R,y^P⦈) in v]
      using ∃ I by fast
```

thus $[\exists\ x\ y\ .\ (|A!,x^P|)$ **&** $(|A!,y^P|)$ **&** $x \neq y$
& $(\boldsymbol{\lambda}^0\ (|R,x^P|)) = (\boldsymbol{\lambda}^0\ (|R,y^P|))\ \mathit{in}\ v]$
using $\exists I$ **by** *fast*
qed

lemma *aclassical2*[*PLM*]:
$[\exists\ x\ y\ .\ (|A!,x^P|)$ **&** $(|A!,y^P|)$ **&** $x \neq y$ **&** $(\forall\ F\ .\ (|F,x^P|) \equiv (|F,y^P|))\ \mathit{in}\ v]$
proof $-$
let $?R_1 = \boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \forall\ F\ .\ (|F,x^P|) \equiv (|F,y^P|))$
have $[\exists\ x\ y\ .\ (|A!,x^P|)$ **&** $(|A!,y^P|)$ **&** $x \neq y$
& $(\boldsymbol{\lambda} z.\ (|?R_1,z^P,x^P|)) = (\boldsymbol{\lambda} z.\ (|?R_1,z^P,y^P|))\ \mathit{in}\ v]$
using *aclassical-1* **by** (*rule* $\forall E$)
then obtain a **where**
$[\exists\ y\ .\ (|A!,a^P|)$ **&** $(|A!,y^P|)$ **&** $a \neq y$
& $(\boldsymbol{\lambda} z.\ (|?R_1,z^P,a^P|)) = (\boldsymbol{\lambda} z.\ (|?R_1,z^P,y^P|))\ \mathit{in}\ v]$
by (*rule* $\exists E$)
then obtain b **where** *ab-prop*:
$[(|A!,a^P|)$ **&** $(|A!,b^P|)$ **&** $a \neq b$
& $(\boldsymbol{\lambda} z.\ (|?R_1,z^P,a^P|)) = (\boldsymbol{\lambda} z.\ (|?R_1,z^P,b^P|))\ \mathit{in}\ v]$
by (*rule* $\exists E$)
have $[(|?R_1,\ a^P,\ a^P|)\ \mathit{in}\ v]$
apply (*rule beta-C-meta-2*[*equiv-rl*])
apply *show-proper*
using *oth-class-taut-4-a*[*THEN* $\forall I$] **by** *fast*
hence $[(|\boldsymbol{\lambda}\ z\ .\ (|?R_1,\ z^P,\ a^P|),\ a^P|)\ \mathit{in}\ v]$
apply $-$ **apply** (*rule beta-C-meta-1*[*equiv-rl*])
apply *show-proper*
by *auto*
hence $[(|\boldsymbol{\lambda}\ z\ .\ (|?R_1,\ z^P,\ b^P|),\ a^P|)\ \mathit{in}\ v]$
using *ab-prop*[*conj2*] *l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *fast*
hence $[(|?R_1,\ a^P,\ b^P|)\ \mathit{in}\ v]$
apply (*safe intro*!: *beta-C-meta-1*[**where** $\varphi=$
$\lambda z\ .\ (|\boldsymbol{\lambda}^2\ (\lambda x\ y.\ \forall F.\ (|F,x^P|) \equiv (|F,y^P|)),z,b^P|)$, *equiv-lr*])
by *show-proper*
moreover have *IsProperInXY* $(\lambda x\ y.\ \forall F.\ (|F,x|) \equiv (|F,y|))$
by *show-proper*
ultimately have $[\forall F.\ (|F,a^P|) \equiv (|F,b^P|)\ \mathit{in}\ v]$
using *beta-C-meta-2*[*equiv-lr*] **by** *blast*
hence $[(|A!,a^P|)$ **&** $(|A!,b^P|)$ **&** $a \neq b$ **&** $(\forall F.\ (|F,a^P|) \equiv (|F,b^P|))\ \mathit{in}\ v]$
using *ab-prop*[*conj1*] **&***I* **by** *presburger*
hence $[\exists\ y\ .\ (|A!,a^P|)$ **&** $(|A!,y^P|)$ **&** $a \neq y$ **&** $(\forall F.\ (|F,a^P|) \equiv (|F,y^P|))\ \mathit{in}\ v]$
using $\exists I$ **by** *fast*
thus *?thesis* **using** $\exists I$ **by** *fast*
qed

9.13 Propositional Properties

lemma *prop-prop2-1*:
$[\forall\ p\ .\ \exists\ F\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)\ \mathit{in}\ v]$
proof (*rule* $\forall I$)
fix p
have $[(\boldsymbol{\lambda}\ x\ .\ p) = (\boldsymbol{\lambda}\ x\ .\ p)\ \mathit{in}\ v]$
using *id-eq-prop-prop-1* **by** *auto*
thus $[\exists\ F\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)\ \mathit{in}\ v]$
by *PLM-solver*
qed

lemma *prop-prop2-2*:
$[F = (\boldsymbol{\lambda}\ x\ .\ p) \rightarrow \square(\forall\ x\ .\ (|F,x^P|) \equiv p)\ \mathit{in}\ v]$
proof (*rule CP*)
assume *1*: $[F = (\boldsymbol{\lambda}\ x\ .\ p)\ \mathit{in}\ v]$
{

fix v
{
fix x
have $[\llparenthesis(\boldsymbol{\lambda}\ x\ .\ p),\ x^P\rrparenthesis \equiv p\ in\ v]$
apply (*rule beta-C-meta-1*)
by *show-proper*
}
hence $[\forall\ \ x\ .\ \llparenthesis(\boldsymbol{\lambda}\ x\ .\ p),\ x^P\rrparenthesis \equiv p\ in\ v]$
by (*rule* $\forall\, I$)
}
hence $[\Box(\forall\ \ x\ .\ \llparenthesis(\boldsymbol{\lambda}\ x\ .\ p),\ x^P\rrparenthesis \equiv p)\ in\ v]$
by (*rule RN*)
thus $[\Box(\forall x.\ \llparenthesis F,x^P\rrparenthesis \equiv p)\ in\ v]$
using *l-identity*[*axiom-instance*,*deduction*,*deduction*,
OF 1[*THEN id-eq-prop-prop-2*[*deduction*]]] **by** *fast*
qed

lemma *prop-prop2-3*:
$[Propositional\ F \rightarrow \Box(Propositional\ F)\ in\ v]$
proof (*rule CP*)
assume $[Propositional\ F\ in\ v]$
hence $[\exists\ \ p\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)\ in\ v]$
unfolding *Propositional-def* **.**
then obtain q **where** $[F = (\boldsymbol{\lambda}\ x\ .\ q)\ in\ v]$
by (*rule* $\exists\, E$)
hence $[\Box(F = (\boldsymbol{\lambda}\ x\ .\ q))\ in\ v]$
using *id-nec*[*equiv-lr*] **by** *auto*
hence $[\exists\ \ p\ .\ \Box(F = (\boldsymbol{\lambda}\ x\ .\ p))\ in\ v]$
using $\exists\, I$ **by** *fast*
thus $[\Box(Propositional\ F)\ in\ v]$
unfolding *Propositional-def*
using *sign-S5-thm-1*[*deduction*] **by** *fast*
qed

lemma *prop-indis*:
$[Indiscriminate\ F \rightarrow (\neg(\exists\ \ x\ y\ .\ \llparenthesis F,x^P\rrparenthesis\ \&\ (\neg\llparenthesis F,y^P\rrparenthesis)))\ in\ v]$
proof (*rule CP*)
assume $[Indiscriminate\ F\ in\ v]$
hence *1*: $[\Box((\exists x.\ \llparenthesis F,x^P\rrparenthesis) \rightarrow (\forall x.\ \llparenthesis F,x^P\rrparenthesis))\ in\ v]$
unfolding *Indiscriminate-def* **.**
{
assume $[\exists\ \ x\ y\ .\ \llparenthesis F,x^P\rrparenthesis\ \&\ \neg\llparenthesis F,y^P\rrparenthesis\ in\ v]$
then obtain x **where** $[\exists\ \ y\ .\ \llparenthesis F,x^P\rrparenthesis\ \&\ \neg\llparenthesis F,y^P\rrparenthesis\ in\ v]$
by (*rule* $\exists\, E$)
then obtain y **where** *2*: $[\llparenthesis F,x^P\rrparenthesis\ \&\ \neg\llparenthesis F,y^P\rrparenthesis\ in\ v]$
by (*rule* $\exists\, E$)
hence $[\exists\ \ x\ .\ \llparenthesis F,\ x^P\rrparenthesis\ in\ v]$
using **&**$E(1)$ $\exists\, I$ **by** *fast*
hence $[\forall\ \ x\ .\ \llparenthesis F,x^P\rrparenthesis\ in\ v]$
using *1*[*THEN qml-2*[*axiom-instance*, *deduction*], *deduction*] **by** *fast*
hence $[\llparenthesis F,y^P\rrparenthesis\ in\ v]$
using *cqt-orig-1*[*deduction*] **by** *fast*
hence $[\llparenthesis F,y^P\rrparenthesis\ \&\ (\neg\llparenthesis F,y^P\rrparenthesis)\ in\ v]$
using *2* **&**I **&**E **by** *fast*
hence $[\neg(\exists\ \ x\ y\ .\ \llparenthesis F,x^P\rrparenthesis\ \&\ \neg\llparenthesis F,y^P\rrparenthesis)\ in\ v]$
using *pl-1*[*axiom-instance*, *deduction*, *THEN modus-tollens-1*]
oth-class-taut-1-a **by** *blast*
}
thus $[\neg(\exists\ \ x\ y\ .\ \llparenthesis F,x^P\rrparenthesis\ \&\ \neg\llparenthesis F,y^P\rrparenthesis)\ in\ v]$
using *reductio-aa-2 if-p-then-p deduction-theorem* **by** *blast*
qed

```
lemma prop-in-thm:
  [Propositional F → Indiscriminate F in v]
  proof (rule CP)
    assume [Propositional F in v]
    hence [□(Propositional F) in v]
      using prop-prop2-3[deduction] by auto
    moreover {
      fix w
      assume [∃ p . (F = (λ y . p)) in w]
      then obtain q where q-prop: [F = (λ y . q) in w]
        by (rule ∃E)
      {
        assume [∃ x . ⦇F,x^P⦈ in w]
        then obtain a where [⦇F,a^P⦈ in w]
          by (rule ∃E)
        hence [⦇λ y . q, a^P⦈ in w]
          using q-prop l-identity[axiom-instance,deduction,deduction] by fast
        hence q: [q in w]
          apply (safe intro!: beta-C-meta-1[where φ=λy. q, equiv-lr])
           apply show-proper
          by simp
        {
          fix x
          have [⦇λ y . q, x^P⦈ in w]
            apply (safe intro!: q beta-C-meta-1[equiv-rl])
            by show-proper
          hence [⦇F,x^P⦈ in w]
            using q-prop[eq-sym] l-identity[axiom-instance, deduction, deduction]
            by fast
        }
        hence [∀ x . ⦇F,x^P⦈ in w]
          by (rule ∀I)
      }
      hence [(∃ x . ⦇F,x^P⦈) → (∀ x . ⦇F, x^P⦈) in w]
        by (rule CP)
    }
    ultimately show [Indiscriminate F in v]
      unfolding Propositional-def Indiscriminate-def
      using RM-1[deduction] deduction-theorem by blast
  qed

lemma prop-in-f-1:
  [Necessary F → Indiscriminate F in v]
  unfolding Necessary-defs Indiscriminate-def
  using pl-1[axiom-instance, THEN RM-1] by simp

lemma prop-in-f-2:
  [Impossible F → Indiscriminate F in v]
  proof –
    {
      fix w
      have [(¬(∃ x . ⦇F,x^P⦈)) → ((∃ x . ⦇F,x^P⦈) → (∀ x . ⦇F,x^P⦈)) in w]
        using useful-tautologies-3 by auto
      hence [(∀ x . ¬⦇F,x^P⦈) → ((∃ x . ⦇F,x^P⦈) → (∀ x . ⦇F,x^P⦈)) in w]
        apply – apply (PLM-subst-method ¬(∃ x. ⦇F,x^P⦈) (∀ x. ¬⦇F,x^P⦈))
        using cqt-further-4 unfolding exists-def by fast+
    }
    thus ?thesis
      unfolding Impossible-defs Indiscriminate-def using RM-1 CP by blast
  qed

lemma prop-in-f-3-a:
```

[¬(*Indiscriminate* (*E*!)) *in v*]
proof (*rule reductio-aa-2*)
show [□¬(∀ *x*. ⦇*E*!,x^P⦈) *in v*]
using *a-objects-exist-3* **.**
next
assume [*Indiscriminate E*! *in v*]
thus [¬□¬(∀ *x* . ⦇*E*!,x^P⦈) *in v*]
unfolding *Indiscriminate-def*
using *o-objects-exist-1 KBasic2-5*[*deduction*,*deduction*]
unfolding *diamond-def* **by** *blast*
qed

lemma *prop-in-f-3-b*:
[¬(*Indiscriminate* ($E!^-$)) *in v*]
proof (*rule reductio-aa-2*)
assume [*Indiscriminate* ($E!^-$) *in v*]
moreover have [□(∃ *x* . ⦇$E!^-$, x^P⦈) *in v*]
apply (*PLM-subst-method* λ *x* . ¬⦇*E*!, x^P⦈ λ *x* . ⦇$E!^-$, x^P⦈)
using *thm-relation-negation-1-1*[*equiv-sym*] **apply** *simp*
unfolding *exists-def*
apply (*PLM-subst-method* λ *x* . ⦇*E*!, x^P⦈ λ *x* . ¬¬⦇*E*!, x^P⦈)
using *oth-class-taut-4-b* **apply** *simp*
using *a-objects-exist-3* **by** *auto*
ultimately have [□(∀ *x*. ⦇$E!^-$,x^P⦈) *in v*]
unfolding *Indiscriminate-def*
using *qml-1*[*axiom-instance*, *deduction*, *deduction*] **by** *blast*
thus [□(∀ *x*. ¬⦇*E*!,x^P⦈) *in v*]
apply −
apply (*PLM-subst-method* λ *x* . ⦇$E!^-$, x^P⦈ λ *x* . ¬⦇*E*!, x^P⦈)
using *thm-relation-negation-1-1* **by** *auto*
next
show [¬□(∀ *x* . ¬⦇*E*!, x^P⦈) *in v*]
using *o-objects-exist-1*
unfolding *diamond-def exists-def*
apply −
apply (*PLM-subst-method* ¬¬(∀ *x*. ¬⦇*E*!,x^P⦈) ∀ *x*. ¬⦇*E*!,x^P⦈)
using *oth-class-taut-4-b*[*equiv-sym*] **by** *auto*
qed

lemma *prop-in-f-3-c*:
[¬(*Indiscriminate* (*O*!)) *in v*]
proof (*rule reductio-aa-2*)
show [¬(∀ *x* . ⦇*O*!,x^P⦈) *in v*]
using *a-objects-exist-2*[*THEN qml-2*[*axiom-instance*, *deduction*]]
by *blast*
next
assume [*Indiscriminate O*! *in v*]
thus [(∀ *x* . ⦇*O*!,x^P⦈) *in v*]
unfolding *Indiscriminate-def*
using *o-objects-exist-2 qml-1*[*axiom-instance*, *deduction*, *deduction*]
qml-2[*axiom-instance*, *deduction*] **by** *blast*
qed

lemma *prop-in-f-3-d*:
[¬(*Indiscriminate* (*A*!)) *in v*]
proof (*rule reductio-aa-2*)
show [¬(∀ *x* . ⦇*A*!,x^P⦈) *in v*]
using *o-objects-exist-3*[*THEN qml-2*[*axiom-instance*, *deduction*]]
by *blast*
next
assume [*Indiscriminate A*! *in v*]
thus [(∀ *x* . ⦇*A*!,x^P⦈) *in v*]
unfolding *Indiscriminate-def*

```
    using a-objects-exist-1 qml-1[axiom-instance, deduction, deduction]
        qml-2[axiom-instance, deduction] by blast
  qed

lemma prop-in-f-4-a:
  [¬(Propositional E!) in v]
  using prop-in-thm[deduction] prop-in-f-3-a modus-tollens-1 CP
  by meson

lemma prop-in-f-4-b:
  [¬(Propositional (E!⁻)) in v]
  using prop-in-thm[deduction] prop-in-f-3-b modus-tollens-1 CP
  by meson

lemma prop-in-f-4-c:
  [¬(Propositional (O!)) in v]
  using prop-in-thm[deduction] prop-in-f-3-c modus-tollens-1 CP
  by meson

lemma prop-in-f-4-d:
  [¬(Propositional (A!)) in v]
  using prop-in-thm[deduction] prop-in-f-3-d modus-tollens-1 CP
  by meson

lemma prop-prop-nec-1:
  [◊(∃ p . F = (λ x . p)) → (∃ p . F = (λ x . p)) in v]
  proof (rule CP)
    assume [◊(∃ p . F = (λ x . p)) in v]
    hence [∃ p . ◊(F = (λ x . p)) in v]
      using BF◊[deduction] by auto
    then obtain p where [◊(F = (λ x . p)) in v]
      by (rule ∃ E)
    hence [◊□(∀ x. ⦃x^P,F⦄ ≡ ⦃x^P,λx. p⦄) in v]
      unfolding identity-defs .
    hence [□(∀ x. ⦃x^P,F⦄ ≡ ⦃x^P,λx. p⦄) in v]
      using 5◊[deduction] by auto
    hence [(F = (λ x . p)) in v]
      unfolding identity-defs .
    thus [∃ p . (F = (λ x . p)) in v]
      by PLM-solver
  qed

lemma prop-prop-nec-2:
  [(∀ p . F ≠ (λ x . p)) → □(∀ p . F ≠ (λ x . p)) in v]
  apply (PLM-subst-method
        ¬(∃  p . (F = (λ x . p)))
        (∀ p . ¬(F = (λ x . p))))
   using cqt-further-4 apply blast
  apply (PLM-subst-method
        ¬◊(∃ p. F = (λx. p))
        □¬(∃ p. F = (λx. p)))
   using KBasic2-4[equiv-sym] prop-prop-nec-1
        contraposition-1 by auto

lemma prop-prop-nec-3:
  [(∃ p . F = (λ x . p)) → □(∃ p . F = (λ x . p)) in v]
  using prop-prop-nec-1 derived-S5-rules-1-b by simp

lemma prop-prop-nec-4:
  [◊(∀ p . F ≠ (λ x . p)) → (∀ p . F ≠ (λ x . p)) in v]
  using prop-prop-nec-2 derived-S5-rules-2-b by simp

lemma enc-prop-nec-1:
```

$[\Diamond(\forall\ F\ .\ \{\!|x^P,\ F|\!\} \rightarrow (\exists\ p\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)))$
$\rightarrow (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \rightarrow (\exists\ p\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)))\ in\ v]$
proof (*rule CP*)
assume $[\Diamond(\forall F.\ \{\!|x^P,F|\!\} \rightarrow (\exists p.\ F = (\boldsymbol{\lambda}x.\ p)))\ in\ v]$
hence *1*: $[(\forall F.\ \Diamond(\{\!|x^P,F|\!\} \rightarrow (\exists p.\ F = (\boldsymbol{\lambda}x.\ p))))\ in\ v]$
using *Buridan*$\Diamond$[*deduction*] **by** *auto*
{
fix Q
assume $[\{\!|x^P,Q|\!\}\ in\ v]$
hence $[\Box\{\!|x^P,Q|\!\}\ in\ v]$
using *encoding*[*axiom-instance*, *deduction*] **by** *auto*
moreover have $[\Diamond(\{\!|x^P,Q|\!\} \rightarrow (\exists p.\ Q = (\boldsymbol{\lambda}x.\ p)))\ in\ v]$
using *cqt-1*[*axiom-instance*, *deduction*] *1* **by** *fast*
ultimately have $[\Diamond(\exists p.\ Q = (\boldsymbol{\lambda}x.\ p))\ in\ v]$
using *KBasic2-9*[*equiv-lr*,*deduction*] **by** *auto*
hence $[(\exists p.\ Q = (\boldsymbol{\lambda}x.\ p))\ in\ v]$
using *prop-prop-nec-1*[*deduction*] **by** *auto*
}
thus $[(\forall\ F\ .\ \{\!|x^P,\ F|\!\} \rightarrow (\exists\ p\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)))\ in\ v]$
apply − **by** *PLM-solver*
qed

lemma *enc-prop-nec-2*:
$[(\forall\ F\ .\ \{\!|x^P,\ F|\!\} \rightarrow (\exists\ p\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p))) \rightarrow \Box(\forall\ F\ .\ \{\!|x^P,\ F|\!\}$
$\rightarrow (\exists\ p\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)))\ in\ v]$
using *derived-S5-rules-1-b enc-prop-nec-1* **by** *blast*
end
end

10 Possible Worlds

locale *PossibleWorlds* = *PLM*
begin

10.1 Definitions

definition *Situation* **where**
Situation $x \equiv (\!|A!,x|\!)$ **&** $(\forall\ F.\ \{\!|x,F|\!\} \rightarrow$ *Propositional* $F)$

definition *EncodeProposition* (**infixl** $\boldsymbol{\Sigma}$ *70*) **where**
$x\boldsymbol{\Sigma}p \equiv (\!|A!,x|\!)$ **&** $\{\!|x,\ \boldsymbol{\lambda}\ x\ .\ p|\!\}$
definition *TrueInSituation* (**infixl** $\models$ *10*) **where**
$x \models p \equiv$ *Situation* x **&** $x\boldsymbol{\Sigma}p$
definition *PossibleWorld* **where**
PossibleWorld $x \equiv$ *Situation* x **&** $\Diamond(\forall\ p\ .\ x\boldsymbol{\Sigma}p \equiv p)$

10.2 Auxiliary Lemmas

lemma *possit-sit-1*:
$[$*Situation* $(x^P) \equiv \Box($*Situation* $(x^P))\ in\ v]$
proof (*rule* $\equiv I$; *rule CP*)
assume $[$*Situation* $(x^P)\ in\ v]$
hence *1*: $[(\!|A!,x^P|\!)$ **&** $(\forall\ F.\ \{\!|x^P,F|\!\} \rightarrow$ *Propositional* $F)\ in\ v]$
unfolding *Situation-def* **by** *auto*
have $[\Box(\!|A!,x^P|\!)\ in\ v]$
using *1*[*conj1*, *THEN oa-facts-2*[*deduction*]] **.**
moreover have $[\Box(\forall\ F.\ \{\!|x^P,F|\!\} \rightarrow$ *Propositional* $F)\ in\ v]$
using *1*[*conj2*] **unfolding** *Propositional-def*
by (*rule enc-prop-nec-2*[*deduction*])
ultimately show $[\Box$*Situation* $(x^P)\ in\ v]$
unfolding *Situation-def*

apply *cut-tac* **apply** (*rule KBasic-3*[*equiv-rl*])
by (*rule intro-elim-1*)
next
assume [$\Box$*Situation* (x^P) *in v*]
thus [*Situation* (x^P) *in v*]
using *qml-2*[*axiom-instance*, *deduction*] **by** *auto*
qed

lemma *possworld-nec*:
[*PossibleWorld* (x^P) $\equiv$ $\Box$(*PossibleWorld* (x^P)) *in v*]
apply (*rule* $\equiv I$; *rule CP*)
subgoal unfolding *PossibleWorld-def*
apply (*rule KBasic-3*[*equiv-rl*])
apply (*rule intro-elim-1*)
using *possit-sit-1*[*equiv-lr*] **&**$E(1)$ **apply** *blast*
using *qml-3*[*axiom-instance*, *deduction*] **&**$E(2)$ **by** *blast*
using *qml-2*[*axiom-instance*,*deduction*] **by** *auto*

lemma *TrueInWorldNecc*:
[((x^P) $\models$ *p*) $\equiv$ $\Box$((x^P) $\models$ *p*) *in v*]
proof (*rule* $\equiv I$; *rule CP*)
assume [x^P $\models$ *p in v*]
hence [*Situation* (x^P) **&** (⦇$A!$,x^P⦈ **&** ⦃x^P,$\boldsymbol{\lambda}x.\ p$⦄) *in v*]
unfolding *TrueInSituation-def EncodeProposition-def* **.**
hence [($\Box$*Situation* (x^P) **&** $\Box$⦇$A!$,x^P⦈) **&** $\Box$⦃x^P, $\boldsymbol{\lambda}x.\ p$⦄ *in v*]
using **&**I **&**E *possit-sit-1*[*equiv-lr*] *oa-facts-2*[*deduction*]
encoding[*axiom-instance*,*deduction*] **by** *metis*
thus [$\Box$((x^P) $\models$ *p*) *in v*]
unfolding *TrueInSituation-def EncodeProposition-def*
using *KBasic-3*[*equiv-rl*] **&**I **&**E **by** *metis*
next
assume [$\Box$(x^P $\models$ *p*) *in v*]
thus [x^P $\models$ *p in v*]
using *qml-2*[*axiom-instance*,*deduction*] **by** *auto*
qed

lemma *PossWorldAux*:
[((⦇$A!$,x^P⦈ **&** ($\forall$ *F* . (⦃x^P,F⦄ $\equiv$ ($\exists$ *p* . *p* **&** (F = ($\boldsymbol{\lambda}$ *x* . *p*))))))
$\rightarrow$ (*PossibleWorld* (x^P)) *in v*]
proof (*rule CP*)
assume *DefX*: [⦇$A!$,x^P⦈ **&** ($\forall$ *F* . (⦃x^P,F⦄ $\equiv$
($\exists$ *p* . *p* **&** (F = ($\boldsymbol{\lambda}$ *x* . *p*))))) *in v*]

have [*Situation* (x^P) *in v*]
proof −
have [⦇$A!$,x^P⦈ *in v*]
using *DefX*[*conj1*] **.**
moreover have [($\forall F$. ⦃x^P,F⦄ $\rightarrow$ *Propositional F*) *in v*]
proof (*rule* $\forall I$; *rule CP*)
fix *F*
assume [⦃x^P,F⦄ *in v*]
moreover have [⦃x^P,F⦄ $\equiv$ ($\exists$ *p* . *p* **&** (F = ($\boldsymbol{\lambda}$ *x* . *p*))) *in v*]
using *DefX*[*conj2*] *cqt-1*[*axiom-instance*, *deduction*] **by** *auto*
ultimately have [($\exists$ *p* . *p* **&** (F = ($\boldsymbol{\lambda}$ *x* . *p*))) *in v*]
using $\equiv E(1)$ **by** *blast*
then obtain *p* **where** [*p* **&** (F = ($\boldsymbol{\lambda}$ *x* . *p*)) *in v*]
by (*rule* $\exists E$)
hence [(F = ($\boldsymbol{\lambda}$ *x* . *p*)) *in v*]
by (*rule* **&**$E(2)$)
hence [($\exists$ *p* . (F = ($\boldsymbol{\lambda}$ *x* . *p*))) *in v*]
by *PLM-solver*
thus [*Propositional F in v*]

```
          unfolding Propositional-def .
      qed
    ultimately show [Situation (x^P) in v]
      unfolding Situation-def by (rule &I)
  qed
  moreover have [◇(∀ p. x^P Σ p ≡ p) in v]
    unfolding EncodeProposition-def
    proof (rule TBasic[deduction]; rule ∀ I)
      fix q
      have EncodeLambda:
        [⦃x^P, λx. q⦄ ≡ (∃ p . p & ((λx. q) = (λ x . p))) in v]
        using DefX[conj2] by (rule cqt-1[axiom-instance, deduction])
      moreover {
         assume [q in v]
         moreover have [(λx. q) = (λ x . q) in v]
          using id-eq-prop-prop-1 by auto
         ultimately have [q & ((λx. q) = (λ x . q)) in v]
           by (rule &I)
         hence [∃ p . p & ((λx. q) = (λ x . p)) in v]
           by PLM-solver
         moreover have [⦇A!,x^P⦈ in v]
           using DefX[conj1] .
         ultimately have [⦇A!,x^P⦈ & ⦃x^P, λx. q⦄ in v]
           using EncodeLambda[equiv-rl] &I by auto
      }
      moreover {
        assume [⦇A!,x^P⦈ & ⦃x^P, λx. q⦄ in v]
        hence [⦃x^P, λx. q⦄ in v]
          using &E(2) by auto
        hence [∃ p . p & ((λx. q) = (λ x . p)) in v]
          using EncodeLambda[equiv-lr] by auto
        then obtain p where p-and-lambda-q-is-lambda-p:
          [p & ((λx. q) = (λ x . p)) in v]
          by (rule ∃ E)
        have [⦇(λ x . p), x^P⦈ ≡ p in v]
          apply (rule beta-C-meta-1)
          by show-proper
        hence [⦇(λ x . p), x^P⦈ in v]
          using p-and-lambda-q-is-lambda-p[conj1] ≡E(2) by auto
        hence [⦇(λ x . q), x^P⦈ in v]
          using p-and-lambda-q-is-lambda-p[conj2, THEN id-eq-prop-prop-2[deduction]]
            l-identity[axiom-instance, deduction, deduction] by fast
        moreover have [⦇(λ x . q), x^P⦈ ≡ q in v]
          apply (rule beta-C-meta-1) by show-proper
        ultimately have [q in v]
          using ≡E(1) by blast
      }
      ultimately show [⦇A!,x^P⦈ & ⦃x^P,λx. q⦄ ≡ q in v]
        using &I ≡I CP by auto
    qed

  ultimately show [PossibleWorld (x^P) in v]
    unfolding PossibleWorld-def by (rule &I)
qed
```

10.3 For every syntactic Possible World there is a semantic Possible World

```
theorem SemanticPossibleWorldForSyntacticPossibleWorlds:
  ∀ x . [PossibleWorld (x^P) in w] ⟶
    (∃ v . ∀ p . [(x^P ⊨ p) in w] ⟷ [p in v])
  proof
    fix x
```

```
{
 assume PossWorldX: [PossibleWorld (x^P) in w]
 hence SituationX: [Situation (x^P) in w]
   unfolding PossibleWorld-def apply cut-tac by PLM-solver
 have PossWorldExpanded:
   [⦇A!,x^P⦈ & (∀ F. ⦃x^P,F⦄ → (∃ p. F = (λx. p)))
     & ◇(∀ p. ⦇A!,x^P⦈ & ⦃x^P,λx. p⦄ ≡ p) in w]
    using PossWorldX
    unfolding PossibleWorld-def Situation-def
              Propositional-def EncodeProposition-def .
 have AbstractX: [⦇A!,x^P⦈ in w]
   using PossWorldExpanded[conj1,conj1] .

 have [◇(∀ p. ⦃x^P,λx. p⦄ ≡ p) in w]
   apply (PLM-subst-method
          λp. ⦇A!,x^P⦈ & ⦃x^P,λx. p⦄
          λ p . ⦃x^P,λx. p⦄)
    subgoal using PossWorldExpanded[conj1,conj1,THEN oa-facts-2[deduction]]
           using Semantics.T6 apply cut-tac by PLM-solver
   using PossWorldExpanded[conj2] .

 hence ∃ v. ∀ p. ([⦃x^P,λx. p⦄ in v])
                 = [p in v]
  unfolding diamond-def equiv-def conj-def
  apply (simp add: Semantics.T4 Semantics.T6 Semantics.T5
                   Semantics.T8)
  by auto

 then obtain v where PropsTrueInSemWorld:
   ∀ p. ([⦃x^P,λx. p⦄ in v]) = [p in v]
   by auto
 {
   fix p
   {
    assume [((x^P) ⊨ p) in w]
    hence [((x^P) ⊨ p) in v]
      using TrueInWorldNecc[equiv-lr] Semantics.T6 by simp
    hence [Situation (x^P) & (⦇A!,x^P⦈ & ⦃x^P,λx. p⦄) in v]
      unfolding TrueInSituation-def EncodeProposition-def .
    hence [⦃x^P,λx. p⦄ in v]
      using &E(2) by blast
    hence [p in v]
      using PropsTrueInSemWorld by blast
   }
   moreover {
    assume [p in v]
    hence [⦃x^P,λx. p⦄ in v]
      using PropsTrueInSemWorld by blast
    hence [(x^P) ⊨ p in v]
      apply cut-tac unfolding TrueInSituation-def EncodeProposition-def
      apply (rule &I) using SituationX[THEN possit-sit-1[equiv-lr]]
      subgoal using Semantics.T6 by auto
      apply (rule &I)
      subgoal using AbstractX[THEN oa-facts-2[deduction]]
        using Semantics.T6 by auto
      by assumption
    hence [□((x^P) ⊨ p) in v]
      using TrueInWorldNecc[equiv-lr] by simp
    hence [(x^P) ⊨ p in w]
      using Semantics.T6 by simp
   }
   ultimately have [p in v] ⟷ [(x^P) ⊨ p in w]
    by auto
```

```
      }
      hence (∃ v . ∀ p . [p in v] ⟷ [(x^P) ⊨ p in w])
        by blast
    }
    thus [PossibleWorld (x^P) in w] ⟶
          (∃ v. ∀ p . [(x^P) ⊨ p in w] ⟷ [p in v])
      by blast
  qed
```

10.4 For every semantic Possible World there is a syntactic Possible World

```
theorem SyntacticPossibleWorldForSemanticPossibleWorlds:
  ∀ v . ∃ x . [PossibleWorld (x^P) in w] ∧
   (∀ p . [p in v] ⟷ [((x^P) ⊨ p) in w])
  proof
    fix v
    have [∃ x. ⦇A!,x^P⦈ & (∀ F . (⦃x^P,F⦄ ≡
          (∃ p . p & (F = (λ x . p))))) in v]
      using A-objects[axiom-instance] by fast
    then obtain x where DefX:
      [⦇A!,x^P⦈ & (∀ F . (⦃x^P,F⦄ ≡ (∃ p . p & (F = (λ x . p))))) in v]
      by (rule ∃E)
    hence PossWorldX: [PossibleWorld (x^P) in v]
      using PossWorldAux[deduction] by blast
    hence [PossibleWorld (x^P) in w]
      using possworld-nec[equiv-lr] Semantics.T6 by auto
    moreover have (∀ p . [p in v] ⟷ [(x^P) ⊨ p in w])
    proof
      fix q
      {
         assume [q in v]
         moreover have [(λ x . q) = (λ x . q) in v]
           using id-eq-prop-prop-1 by auto
         ultimately have [q & (λ x . q) = (λ x . q) in v]
           using &I by auto
         hence [(∃ p . p & ((λ x . q) = (λ x . p))) in v]
           by PLM-solver
         hence 4: [⦃x^P, (λ x . q)⦄ in v]
           using cqt-1[axiom-instance,deduction, OF DefX[conj2], equiv-rl]
           by blast
         have [(x^P ⊨ q) in v]
           unfolding TrueInSituation-def apply (rule &I)
            using PossWorldX unfolding PossibleWorld-def
            using &E(1) apply blast
           unfolding EncodeProposition-def apply (rule &I)
            using DefX[conj1] apply simp
           using 4 .
        hence [(x^P ⊨ q) in w]
          using TrueInWorldNecc[equiv-lr] Semantics.T6 by auto
      }
      moreover {
        assume [(x^P ⊨ q) in w]
        hence [(x^P ⊨ q) in v]
           using TrueInWorldNecc[equiv-lr] Semantics.T6
           by auto
        hence [⦃x^P, (λ x . q)⦄ in v]
          unfolding TrueInSituation-def EncodeProposition-def
          using &E(2) by blast
        hence [(∃ p . p & ((λ x . q) = (λ x . p))) in v]
          using cqt-1[axiom-instance,deduction, OF DefX[conj2], equiv-lr]
          by blast
        then obtain p where 4:
```

[(p **&** (($\boldsymbol{\lambda}$ x . q) = ($\boldsymbol{\lambda}$ x . p))) *in* v]
by (*rule* ∃ *E*)
have [⦇($\boldsymbol{\lambda}$ x . p),x^P⦈ ≡ p *in* v]
apply (*rule beta-C-meta-1*)
by *show-proper*
hence [⦇($\boldsymbol{\lambda}$ x . q),x^P⦈ ≡ p *in* v]
using *l-identity*[**where** β=($\boldsymbol{\lambda}$ x . q) **and** α=($\boldsymbol{\lambda}$ x . p),
axiom-instance, *deduction*, *deduction*]
using *4*[*conj2*,*THEN id-eq-prop-prop-2*[*deduction*]] **by** *meson*
hence [⦇($\boldsymbol{\lambda}$ x . q),x^P⦈ *in* v] **using** *4*[*conj1*] ≡*E*(*2*) **by** *blast*
moreover have [⦇($\boldsymbol{\lambda}$ x . q),x^P⦈ ≡ q *in* v]
apply (*rule beta-C-meta-1*)
by *show-proper*
ultimately have [q *in* v]
using ≡*E*(*1*) **by** *blast*
}
ultimately show [q *in* v] ⟷ [(x^P) ⊨ q *in* w]
by *blast*
qed
ultimately show ∃ x . [*PossibleWorld* (x^P) *in* w]
∧ (∀ p . [p *in* v] ⟷ [(x^P) ⊨ p *in* w])
by *auto*
qed
end

11 Artificial Theorems

Remark 22. *Some examples of theorems that can be derived from the model structure, but which are not derivable from the deductive system PLM itself.*

locale *ArtificialTheorems*
begin

lemma *lambda-enc-1*:
[⦇$\boldsymbol{\lambda}$$x$. ⦃x^P, F⦄ ≡ ⦃x^P, F⦄, y^P⦈ *in* v]
by (*auto simp*: *meta-defs meta-aux conn-defs forall*-Π_1-*def*)

lemma *lambda-enc-2*:
[⦇$\boldsymbol{\lambda}$ x . ⦃y^P, G⦄, x^P⦈ ≡ ⦃y^P, G⦄ *in* v]
by (*auto simp*: *meta-defs meta-aux conn-defs forall*-Π_1-*def*)

Remark 23. *The following is* not *a theorem and nitpick can find a countermodel. This is expected and important. If this were a theorem, the theory would become inconsistent.*

lemma *lambda-enc-3*:
[(⦇$\boldsymbol{\lambda}$ x . ⦃x^P, F⦄, x^P⦈ → ⦃x^P, F⦄) *in* v]
apply (*simp add*: *meta-defs meta-aux conn-defs forall*-Π_1-*def*)
nitpick[*user-axioms*, *expect*=*genuine*]
oops — countermodel by nitpick

Remark 24. *Instead the following two statements hold.*

lemma *lambda-enc-4*:
[⦇($\boldsymbol{\lambda}$ x . ⦃x^P, F⦄), x^P⦈ *in* v] = (∃ y . $\nu\upsilon$ y = $\nu\upsilon$ x ∧ [⦃y^P, F⦄ *in* v])
by (*simp add*: *meta-defs meta-aux*)

lemma *lambda-ex*:
[⦇($\boldsymbol{\lambda}$ x . φ (x^P)), x^P⦈ *in* v] = (∃ y . $\nu\upsilon$ y = $\nu\upsilon$ x ∧ [φ (y^P) *in* v])
by (*simp add*: *meta-defs meta-aux*)

Remark 25. *These statements can be translated to statements in the embedded logic.*

```
lemma lambda-ex-emb:
  [(|(λ x . φ (x^P)), x^P|) ≡ (∃ y . (∀ F . (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P)) in v]
  proof(rule MetaSolver.EquivI)
    interpret MetaSolver .
    {
      assume [(|(λ x . φ (x^P)), x^P|) in v]
      then obtain y where νυ y = νυ x ∧ [φ (y^P) in v]
        using lambda-ex by blast
      moreover hence [(∀ F . (|F,x^P|) ≡ (|F,y^P|)) in v]
        apply − apply meta-solver
        by (simp add: Semantics.d_κ-proper Semantics.ex1-def)
      ultimately have [∃ y . (∀ F . (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P) in v]
        using ExIRule ConjI by fast
    }
    moreover {
      assume [∃ y . (∀ F . (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P) in v]
      then obtain y where y-def: [(∀ F . (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P) in v]
        by (rule ExERule)
      hence ⋀ F . [(|F,x^P|) in v] = [(|F,y^P|) in v]
        apply − apply (drule ConjE) apply (drule conjunct1)
        apply (drule AllE) apply (drule EquivE) by simp
      hence [(|makeΠ₁ (λ u s w . νυ y = u),x^P|) in v]
          = [(|makeΠ₁ (λ u s w . νυ y = u),y^P|) in v] by auto
      hence νυ y = νυ x by (simp add: meta-defs meta-aux)
      moreover have [φ (y^P) in v] using y-def ConjE by blast
      ultimately have [(|(λ x . φ (x^P)), x^P|) in v]
        using lambda-ex by blast
    }
    ultimately show [(|λx. φ (x^P),x^P|) in v]
        = [∃ y. (∀ F. (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P) in v]
      by auto
  qed

lemma lambda-enc-emb:
  [(|(λ x . {|x^P, F|}), x^P|) ≡ (∃ y . (∀ F . (|F,x^P|) ≡ (|F,y^P|)) & {|y^P, F|}) in v]
  using lambda-ex-emb by fast
```

Remark 26. *In the case of proper maps, the generalized β-conversion reduces to classical β-conversion.*

```
lemma proper-beta:
  assumes IsProperInX φ
  shows [(∃ y . (∀ F . (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P)) ≡ φ (x^P) in v]
proof (rule MetaSolver.EquivI; rule)
  interpret MetaSolver .
  assume [∃ y. (∀ F. (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P) in v]
  then obtain y where y-def: [(∀ F. (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P) in v] by (rule ExERule)
  hence [(|makeΠ₁ (λ u s w . νυ y = u), x^P|) in v] = [(|makeΠ₁ (λ u s w . νυ y = u), y^P|) in v]
    using EquivS AllE ConjE by blast
  hence νυ y = νυ x by (simp add: meta-defs meta-aux)
  thus [φ (x^P) in v]
    using y-def[THEN ConjE[THEN conjunct2]]
          assms IsProperInX.rep-eq valid-in.rep-eq
    by blast
next
  interpret MetaSolver .
  assume [φ (x^P) in v]
  moreover have [∀ F. (|F,x^P|) ≡ (|F,x^P|) in v] apply meta-solver by blast
  ultimately show [∃ y. (∀ F. (|F,x^P|) ≡ (|F,y^P|)) & φ (y^P) in v]
    by (meson ConjI ExI)
qed
```

Remark 27. *The following theorem is a consequence of the constructed Aczel-model, but not part of PLM. Separate research on possible modifications of the embedding suggest that this* *artificial theorem can be avoided by introducing a dependency on states for the mapping from abstract objects to special urelements.*

lemma *lambda-rel-extensional*:
 assumes $[\forall F . (|F,a^P|) \equiv (|F,b^P|)\ in\ v]$
 shows $(\boldsymbol{\lambda}x. (|R,x^P,a^P|)) = (\boldsymbol{\lambda}x . (|R,x^P, b^P|))$
proof −
 interpret *MetaSolver* **.**
 obtain F **where** *F-def*: $F = make\Pi_1\ (\lambda\ u\ s\ w\ .\ u = \nu\upsilon\ a)$ **by** *auto*
 have $[(|F, a^P|) \equiv (|F, b^P|)\ in\ v]$ **using** *assms* **by** (*rule AllE*)
 moreover have $[(|F, a^P|)\ in\ v]$
  unfolding *F-def* **by** (*simp add*: *meta-defs meta-aux*)
 ultimately have $[(|F, b^P|)\ in\ v]$ **using** *EquivE* **by** *auto*
 hence $\nu\upsilon\ a = \nu\upsilon\ b$ **using** *F-def* **by** (*simp add*: *meta-defs meta-aux*)
 thus *?thesis* **by** (*simp add*: *meta-defs meta-aux*)
qed

end

12 Sanity Tests

locale *SanityTests*
begin
 interpretation *MetaSolver*.
 interpretation *Semantics*.

12.1 Consistency

lemma *True*
 nitpick[*expect=genuine*, *user-axioms*, *satisfy*]
 by *auto*

12.2 Intensionality

lemma $[(\boldsymbol{\lambda}y. (q \vee \neg q)) = (\boldsymbol{\lambda}y. (p \vee \neg p))\ in\ v]$
 unfolding *identity*-Π_1-*def conn-defs*
 apply (*rule* Eq_1I) **apply** (*simp add*: *meta-defs*)
 nitpick[*expect = genuine*, *user-axioms=true*, *card i = 2*,
   card j = 2, *card* ω = *1*, *card* σ = *1*,
   sat-solver = MiniSat-JNI, *verbose*, *show-all*]
 oops — Countermodel by Nitpick
lemma $[(\boldsymbol{\lambda}y. (p \vee q)) = (\boldsymbol{\lambda}y. (q \vee p))\ in\ v]$
 unfolding *identity*-Π_1-*def*
 apply (*rule* Eq_1I) **apply** (*simp add*: *meta-defs*)
 nitpick[*expect = genuine*, *user-axioms=true*,
   sat-solver = MiniSat-JNI, *card i = 2*,
   card j = 2, *card* σ = *1*, *card* ω = *1*,
   card υ = *2*, *verbose*, *show-all*]
 oops — Countermodel by Nitpick

12.3 Concreteness coindices with Object Domains

lemma *OrdCheck*:
 $[(|\boldsymbol{\lambda}\ x\ .\ \neg\Box(\neg(|E!, x^P|)), x|)\ in\ v] \longleftrightarrow$
  $(proper\ x) \wedge (case\ (rep\ x)\ of\ \omega\nu\ y \Rightarrow True \mid \text{-} \Rightarrow False)$
 using *OrdinaryObjectsPossiblyConcreteAxiom*
 apply (*simp add*: *meta-defs meta-aux split*: ν.*split* υ.*split*)
 using $\nu\upsilon$-$\omega\nu$-*is*-$\omega\upsilon$ **by** *fastforce*
lemma *AbsCheck*:

$[(\!|\boldsymbol{\lambda}\ x\ .\ \Box(\neg(\!|E!,\ x^P|\!)),\ x|\!)\ \mathit{in}\ v] \longleftrightarrow$
$(\mathit{proper}\ x) \wedge (\mathit{case}\ (\mathit{rep}\ x)\ \mathit{of}\ \alpha\nu\ y \Rightarrow \mathit{True}\ |\ \text{-} \Rightarrow \mathit{False})$
using *OrdinaryObjectsPossiblyConcreteAxiom*
apply (*simp add*: *meta-defs meta-aux split*: ν*.split* υ*.split*)
using *no-*$\alpha\omega$ **by** *blast*

12.4 Justification for Meta-Logical Axioms

Remark 28. *OrdinaryObjectsPossiblyConcreteAxiom is equivalent to "all ordinary objects are possibly concrete".*

lemma *OrdAxiomCheck*:
OrdinaryObjectsPossiblyConcrete $\longleftrightarrow$
$(\forall\ x.\ ([(\!|\boldsymbol{\lambda}\ x\ .\ \neg\Box(\neg(\!|E!,\ x^P|\!)),\ x^P|\!)\ \mathit{in}\ v]$
$\longleftrightarrow (\mathit{case}\ x\ \mathit{of}\ \omega\nu\ y \Rightarrow \mathit{True}\ |\ \text{-} \Rightarrow \mathit{False})))$
unfolding *Concrete-def*
apply (*simp add*: *meta-defs meta-aux split*: ν*.split* υ*.split*)
using $\nu\upsilon$*-*$\omega\nu$*-is-*$\omega\upsilon$ **by** *fastforce*

Remark 29. *OrdinaryObjectsPossiblyConcreteAxiom is equivalent to "all abstract objects are necessarily not concrete".*

lemma *AbsAxiomCheck*:
OrdinaryObjectsPossiblyConcrete $\longleftrightarrow$
$(\forall\ x.\ ([(\!|\boldsymbol{\lambda}\ x\ .\ \Box(\neg(\!|E!,\ x^P|\!)),\ x^P|\!)\ \mathit{in}\ v]$
$\longleftrightarrow (\mathit{case}\ x\ \mathit{of}\ \alpha\nu\ y \Rightarrow \mathit{True}\ |\ \text{-} \Rightarrow \mathit{False})))$
apply (*simp add*: *meta-defs meta-aux split*: ν*.split* υ*.split*)
using $\nu\upsilon$*-*$\omega\nu$*-is-*$\omega\upsilon$ *no-*$\alpha\omega$ **by** *fastforce*

Remark 30. *PossiblyContingentObjectExistsAxiom is equivalent to the corresponding statement in the embedded logic.*

lemma *PossiblyContingentObjectExistsCheck*:
PossiblyContingentObjectExists $\longleftrightarrow [\neg(\Box(\forall\ x.\ (\!|E!,x^P|\!) \to \Box(\!|E!,x^P|\!)))\ \mathit{in}\ v]$
apply (*simp add*: *meta-defs forall-*ν*-def meta-aux split*: ν*.split* υ*.split*)
by (*metis* ν*.simps*(*5*) $\nu\upsilon$*-def* υ*.simps*(*1*) *no-*$\sigma\omega$ υ*.exhaust*)

Remark 31. *PossiblyNoContingentObjectExistsAxiom is equivalent to the corresponding statement in the embedded logic.*

lemma *PossiblyNoContingentObjectExistsCheck*:
PossiblyNoContingentObjectExists $\longleftrightarrow [\neg(\Box(\neg(\forall\ x.\ (\!|E!,x^P|\!) \to \Box(\!|E!,x^P|\!))))\ \mathit{in}\ v]$
apply (*simp add*: *meta-defs forall-*ν*-def meta-aux split*: ν*.split* υ*.split*)
using $\nu\upsilon$*-*$\omega\nu$*-is-*$\omega\upsilon$ **by** *blast*

12.5 Relations in the Meta-Logic

Remark 32. *Material equality in the embedded logic corresponds to equality in the actual state in the meta-logic.*

lemma *mat-eq-is-eq-dj*:
$[\forall\ x\ .\ \Box((\!|F,x^P|\!) \equiv (\!|G,x^P|\!))\ \mathit{in}\ v] \longleftrightarrow$
$((\lambda\ x\ .\ (\mathit{eval}\Pi_1\ F)\ x\ \mathit{dj}) = (\lambda\ x\ .\ (\mathit{eval}\Pi_1\ G)\ x\ \mathit{dj}))$
proof
assume *1*: $[\forall x.\ \Box((\!|F,x^P|\!) \equiv (\!|G,x^P|\!))\ \mathit{in}\ v]$
{
fix v
fix y
obtain x **where** *y-def*: $y = \nu\upsilon\ x$
by (*meson* $\nu\upsilon$*-surj surj-def*)
have $(\exists\, r\ o_1.\ \mathit{Some}\ r = d_1\ F \wedge \mathit{Some}\ o_1 = d_\kappa\ (x^P) \wedge o_1 \in \mathit{ex1}\ r\ v) =$
$(\exists\, r\ o_1.\ \mathit{Some}\ r = d_1\ G \wedge \mathit{Some}\ o_1 = d_\kappa\ (x^P) \wedge o_1 \in \mathit{ex1}\ r\ v)$

using *1* **apply** − **by** *meta-solver*
moreover obtain r **where** *r-def*: $Some\ r = d_1\ F$
unfolding d_1*-def* **by** *auto*
moreover obtain s **where** *s-def*: $Some\ s = d_1\ G$
unfolding d_1*-def* **by** *auto*
moreover have $Some\ x = d_\kappa\ (x^P)$
using d_κ*-proper* **by** *simp*
ultimately have $(x \in ex1\ r\ v) = (x \in ex1\ s\ v)$
by (*metis option.inject*)
hence $(eval\Pi_1\ F)\ y\ dj\ v = (eval\Pi_1\ G)\ y\ dj\ v$
using *r-def s-def y-def* **by** (*simp add*: d_1*.rep-eq ex1-def*)
}
thus $(\lambda x.\ eval\Pi_1\ F\ x\ dj) = (\lambda x.\ eval\Pi_1\ G\ x\ dj)$
by *auto*
next
assume *1*: $(\lambda x.\ eval\Pi_1\ F\ x\ dj) = (\lambda x.\ eval\Pi_1\ G\ x\ dj)$
{
fix $y\ v$
obtain x **where** *x-def*: $x = \nu\upsilon\ y$
by *simp*
hence $eval\Pi_1\ F\ x\ dj = eval\Pi_1\ G\ x\ dj$
using *1* **by** *metis*
moreover obtain r **where** *r-def*: $Some\ r = d_1\ F$
unfolding d_1*-def* **by** *auto*
moreover obtain s **where** *s-def*: $Some\ s = d_1\ G$
unfolding d_1*-def* **by** *auto*
ultimately have $(y \in ex1\ r\ v) = (y \in ex1\ s\ v)$
by (*simp add*: d_1*.rep-eq ex1-def* $\nu\upsilon$*-surj x-def*)
hence $[(\!|F,y^P|\!) \equiv (\!|G,y^P|\!)\ in\ v]$
apply − **apply** *meta-solver*
using *r-def s-def* **by** (*metis Semantics.*d_κ*-proper option.inject*)
}
thus $[\forall x.\ \Box((\!|F,x^P|\!) \equiv (\!|G,x^P|\!))\ in\ v]$
using *T6 T8* **by** *fast*
qed

Remark 33. *Materially equivalent relations are equal in the embedded logic if and only if they also coincide in all other states.*

lemma *mat-eq-is-eq-if-eq-forall-j*:
assumes $[\forall\ x\ .\ \Box((\!|F,x^P|\!) \equiv (\!|G,x^P|\!))\ in\ v]$
shows $[F = G\ in\ v] \longleftrightarrow$
$(\forall\ s\ .\ s \neq dj \longrightarrow (\forall\ x\ .\ (eval\Pi_1\ F)\ x\ s = (eval\Pi_1\ G)\ x\ s))$
proof
interpret *MetaSolver* **.**
assume $[F = G\ in\ v]$
hence $F = G$
apply − **unfolding** *identity*-Π_1*-def* **by** *meta-solver*
thus $\forall s.\ s \neq dj \longrightarrow (\forall x.\ eval\Pi_1\ F\ x\ s = eval\Pi_1\ G\ x\ s)$
by *auto*
next
interpret *MetaSolver* **.**
assume $\forall s.\ s \neq dj \longrightarrow (\forall x.\ eval\Pi_1\ F\ x\ s = eval\Pi_1\ G\ x\ s)$
moreover have $((\lambda\ x\ .\ (eval\Pi_1\ F)\ x\ dj) = (\lambda\ x\ .\ (eval\Pi_1\ G)\ x\ dj))$
using *assms mat-eq-is-eq-dj* **by** *auto*
ultimately have $\forall s\ x.\ eval\Pi_1\ F\ x\ s = eval\Pi_1\ G\ x\ s$
by *metis*
hence $eval\Pi_1\ F = eval\Pi_1\ G$
by *blast*
hence $F = G$
by (*metis eval*Π_1*-inverse*)
thus $[F = G\ in\ v]$
unfolding *identity*-Π_1*-def* **using** Eq_1I **by** *auto*
qed

Remark 34. *Under the assumption that all properties behave in all states like in the actual state the defined equality degenerates to material equality.*

lemma assumes $\forall\ F\ x\ s\ .\ (eval\Pi_1\ F)\ x\ s = (eval\Pi_1\ F)\ x\ dj$
shows $[\forall\ x\ .\ \Box((\!|F,x^P|\!) \equiv (\!|G,x^P|\!))\ in\ v] \longleftrightarrow [F = G\ in\ v]$
by (*metis* (*no-types*) *MetaSolver.Eq*$_1$*S assms identity-*Π_1*-def mat-eq-is-eq-dj mat-eq-is-eq-if-eq-forall-j*)

12.6 Lambda Expressions

lemma *lambda-interpret-1*:
assumes $[a = b\ in\ v]$
shows $(\boldsymbol{\lambda}x.\ (\!|R,x^P,a|\!)) = (\boldsymbol{\lambda}x\ .\ (\!|R,x^P,\ b|\!))$
proof −
have $a = b$
using *MetaSolver.Eq*κ*S Semantics.d*$_\kappa$*-inject assms identity-*κ*-def* **by** *auto*
thus *?thesis* **by** *simp*
qed

lemma *lambda-interpret-2*:
assumes $[a = (\boldsymbol{\iota}y.\ (\!|G,y^P|\!))\ in\ v]$
shows $(\boldsymbol{\lambda}x.\ (\!|R,x^P,a|\!)) = (\boldsymbol{\lambda}x\ .\ (\!|R,x^P,\ \boldsymbol{\iota}y.\ (\!|G,y^P|\!)|\!))$
proof −
have $a = (\boldsymbol{\iota}y.\ (\!|G,y^P|\!))$
using *MetaSolver.Eq*κ*S Semantics.d*$_\kappa$*-inject assms identity-*κ*-def* **by** *auto*
thus *?thesis* **by** *simp*
qed

end

theory *TAO-99-Paradox*
imports *TAO-9-PLM TAO-98-ArtificialTheorems*
begin

13 Paradox

Under the additional assumption that expressions of the form $\lambda x.\ (\!|G,\boldsymbol{\iota}y.\ \varphi\ y\ x|\!)$ for arbitrary φ are *proper maps*, for which β-conversion holds, the theory becomes inconsistent.

13.1 Auxiliary Lemmas

lemma *exe-impl-exists*:
$[(\!|(\boldsymbol{\lambda}x\ .\ \forall\ p\ .\ p \rightarrow p),\ \boldsymbol{\iota}y\ .\ \varphi\ y\ x|\!) \equiv (\exists\,!y\ .\ \boldsymbol{\mathcal{A}}\varphi\ y\ x)\ in\ v]$
proof (*rule* $\equiv I$; *rule CP*)
fix $\varphi :: \nu{\Rightarrow}\nu{\Rightarrow}o$ **and** $x :: \nu$ **and** $v :: i$
assume $[(\!|(\boldsymbol{\lambda}x\ .\ \forall\ p\ .\ p \rightarrow p),\boldsymbol{\iota}y\ .\ \varphi\ y\ x|\!)\ in\ v]$
hence $[\exists\,y.\ \boldsymbol{\mathcal{A}}\varphi\ y\ x\ \&\ (\forall\,z.\ \boldsymbol{\mathcal{A}}\varphi\ z\ x \rightarrow z = y)\ \&\ (\!|(\boldsymbol{\lambda}x\ .\ \forall\ p\ .\ p \rightarrow p),\ y^P|\!)\ in\ v]$
using *nec-russell-axiom*[*equiv-lr*] *SimpleExOrEnc.intros* **by** *auto*
then obtain y **where**
$[\boldsymbol{\mathcal{A}}\varphi\ y\ x\ \&\ (\forall\,z.\ \boldsymbol{\mathcal{A}}\varphi\ z\ x \rightarrow z = y)\ \&\ (\!|(\boldsymbol{\lambda}x\ .\ \forall\ p\ .\ p \rightarrow p),\ y^P|\!)\ in\ v]$
by (*rule Instantiate*)
hence $[\boldsymbol{\mathcal{A}}\varphi\ y\ x\ \&\ (\forall\,z.\ \boldsymbol{\mathcal{A}}\varphi\ z\ x \rightarrow z = y)\ in\ v]$
using $\&E$ **by** *blast*
hence $[\exists\,y\ .\ \boldsymbol{\mathcal{A}}\varphi\ y\ x\ \&\ (\forall\,z.\ \boldsymbol{\mathcal{A}}\varphi\ z\ x \rightarrow z = y)\ in\ v]$
by (*rule existential*)
thus $[\exists\,!y.\ \boldsymbol{\mathcal{A}}\varphi\ y\ x\ in\ v]$
unfolding *exists-unique-def* **by** *simp*
next

fix $\varphi :: \nu{\Rightarrow}\nu{\Rightarrow}\mathrm{o}$ **and** $x :: \nu$ **and** $v :: i$
assume $[\exists\,!y.\ \boldsymbol{\mathcal{A}}\varphi\ y\ x\ in\ v]$
hence $[\exists\, y.\ \boldsymbol{\mathcal{A}}\varphi\ y\ x\ \&\ (\forall\, z.\ \boldsymbol{\mathcal{A}}\varphi\ z\ x \rightarrow z = y)\ in\ v]$
unfolding *exists-unique-def* **by** *simp*
then obtain y **where**
$[\boldsymbol{\mathcal{A}}\varphi\ y\ x\ \&\ (\forall\, z.\ \boldsymbol{\mathcal{A}}\varphi\ z\ x \rightarrow z = y)\ in\ v]$
by (*rule Instantiate*)
moreover have $[(\!|(\boldsymbol{\lambda} x\ .\ \forall\ \ p\ .\ p \rightarrow p), y^P|\!)\ in\ v]$
apply (*rule beta-C-meta-1*[*equiv-rl*])
apply *show-proper*
by *PLM-solver*
ultimately have $[\boldsymbol{\mathcal{A}}\varphi\ y\ x\ \&\ (\forall\, z.\ \boldsymbol{\mathcal{A}}\varphi\ z\ x \rightarrow z = y)$
$\&\ (\!|(\boldsymbol{\lambda} x\ .\ \forall\ \ p\ .\ p \rightarrow p), y^P|\!)\ in\ v]$
using &I **by** *blast*
hence $[\exists\ \ y\ .\ \boldsymbol{\mathcal{A}}\varphi\ y\ x\ \&\ (\forall\, z.\ \boldsymbol{\mathcal{A}}\varphi\ z\ x \rightarrow z = y)$
$\&\ (\!|(\boldsymbol{\lambda} x\ .\ \forall\ \ p\ .\ p \rightarrow p), y^P|\!)\ in\ v]$
by (*rule existential*)
thus $[(\!|(\boldsymbol{\lambda} x\ .\ \forall\ \ p\ .\ p \rightarrow p),\ \boldsymbol{\iota} y.\ \varphi\ y\ x|\!)\ in\ v]$
using *nec-russell-axiom*[*equiv-rl*]
SimpleExOrEnc.intros **by** *auto*
qed

lemma *exists-unique-actual-equiv*:
$[(\exists\,!y\ .\ \boldsymbol{\mathcal{A}}(y = x\ \&\ \psi\ (x^P))) \equiv \boldsymbol{\mathcal{A}}\psi\ (x^P)\ in\ v]$
proof (*rule* $\equiv I$; *rule CP*)
fix $x\ v$
let $?\varphi = \lambda\ y\ x.\ y = x\ \&\ \psi\ (x^P)$
assume $[\exists\,!y.\ \boldsymbol{\mathcal{A}}?\varphi\ y\ x\ in\ v]$
hence $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}?\varphi\ \alpha\ x\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}?\varphi\ \beta\ x \rightarrow \beta = \alpha)\ in\ v]$
unfolding *exists-unique-def* **by** *simp*
then obtain α **where**
$[\boldsymbol{\mathcal{A}}?\varphi\ \alpha\ x\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}?\varphi\ \beta\ x \rightarrow \beta = \alpha)\ in\ v]$
by (*rule Instantiate*)
hence $[\boldsymbol{\mathcal{A}}(\alpha = x\ \&\ \psi\ (x^P))\ in\ v]$
using &E **by** *blast*
thus $[\boldsymbol{\mathcal{A}}(\psi\ (x^P))\ in\ v]$
using *Act-Basic-2*[*equiv-lr*] &E **by** *blast*
next
fix $x\ v$
let $?\varphi = \lambda\ y\ x.\ y = x\ \&\ \psi\ (x^P)$
assume *1*: $[\boldsymbol{\mathcal{A}}\psi\ (x^P)\ in\ v]$
have $[x = x\ in\ v]$
using *id-eq-1*[**where** $'a{=}\nu$] **by** *simp*
hence $[\boldsymbol{\mathcal{A}}(x = x)\ in\ v]$
using *id-act-3*[*equiv-lr*] **by** *fast*
hence $[\boldsymbol{\mathcal{A}}(x = x\ \&\ \psi\ (x^P))\ in\ v]$
using *1 Act-Basic-2*[*equiv-rl*] &I **by** *blast*
hence $[\boldsymbol{\mathcal{A}}?\varphi\ x\ x\ in\ v]$
by *simp*
moreover have $[\forall\,\beta.\ \boldsymbol{\mathcal{A}}?\varphi\ \beta\ x \rightarrow \beta = x\ in\ v]$
proof (*rule* $\forall I$; *rule CP*)
fix β
assume $[\boldsymbol{\mathcal{A}}?\varphi\ \beta\ x\ in\ v]$
hence $[\boldsymbol{\mathcal{A}}(\beta = x)\ in\ v]$
using *Act-Basic-2*[*equiv-lr*] &E **by** *fast*
thus $[\beta = x\ in\ v]$ **using** *id-act-3*[*equiv-rl*] **by** *fast*
qed
ultimately have $[\boldsymbol{\mathcal{A}}?\varphi\ x\ x\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}?\varphi\ \beta\ x \rightarrow \beta = x)\ in\ v]$
using &I **by** *fast*
hence $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}?\varphi\ \alpha\ x\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}?\varphi\ \beta\ x \rightarrow \beta = \alpha)\ in\ v]$
by (*rule existential*)
thus $[\exists\,!y.\ \boldsymbol{\mathcal{A}}?\varphi\ y\ x\ in\ v]$
unfolding *exists-unique-def* **by** *simp*

qed

13.2 Fake β-Conversion using Description Backdoor

definition *backdoor* **where**
backdoor $\equiv \lambda\ \psi\ .\ \boldsymbol{\lambda} x\ .\ (\!|(\boldsymbol{\lambda} x\ .\ \forall\ p\ .\ p \rightarrow p),\ \boldsymbol{\iota} y\ .\ y = x\ \&\ \psi\ (x^P)|\!)$

lemma *fake-beta*:
assumes $\bigwedge\ G\ \varphi.\ IsProperInX\ (\lambda x\ .\ (\!|G,\boldsymbol{\iota} y\ .\ \varphi\ y\ x|\!))$
shows $[(\!|backdoor\ (\lambda\ x\ .\ \psi\ x),\ x^P|\!) \equiv \boldsymbol{\mathcal{A}}\psi\ (x^P)\ in\ v]$
proof (*rule* $\equiv I$; *rule CP*)
assume $[(\!|backdoor\ \psi,x^P|\!)\ in\ v]$
hence $[(\!|\boldsymbol{\lambda} x.\ \forall\, p.\ p \rightarrow p,\boldsymbol{\iota} y.\ y = x\ \&\ \psi\ (x^P)|\!)\ in\ v]$
using *beta-C-meta-1*[*equiv-lr*, *OF assms*]
unfolding *backdoor-def identity-ν-def* **by** *fast*
hence $[\exists\,!y.\ \boldsymbol{\mathcal{A}}\ (y = x\ \&\ \psi\ (x^P))\ in\ v]$
using *exe-impl-exists*[*equiv-lr*] **by** *fast*
thus $[\boldsymbol{\mathcal{A}}\psi\ (x^P)\ in\ v]$
using *exists-unique-actual-equiv*[*equiv-lr*] **by** *blast*
next
assume $[\boldsymbol{\mathcal{A}}\psi\ (x^P)\ in\ v]$
hence $[\exists\,!y.\ \boldsymbol{\mathcal{A}}\ (y = x\ \&\ \psi\ (x^P))\ in\ v]$
using *exists-unique-actual-equiv*[*equiv-rl*] **by** *blast*
hence $[(\!|\boldsymbol{\lambda} x.\ \forall\, p.\ p \rightarrow p,\boldsymbol{\iota} y.\ y = x\ \&\ \psi\ (x^P)|\!)\ in\ v]$
using *exe-impl-exists*[*equiv-rl*] **by** *fast*
thus $[(\!|backdoor\ \psi,x^P|\!)\ in\ v]$
using *beta-C-meta-1*[*equiv-rl*, *OF assms*]
unfolding *backdoor-def* **unfolding** *identity-ν-def* **by** *fast*
qed

lemma *fake-beta-act*:
assumes $\bigwedge\ G\ \varphi.\ IsProperInX\ (\lambda x\ .\ (\!|G,\boldsymbol{\iota} y\ .\ \varphi\ y\ x|\!))$
shows $[(\!|backdoor\ (\lambda\ x\ .\ \psi\ x),\ x^P|\!) \equiv \psi\ (x^P)\ in\ dw]$
using *fake-beta*[*OF assms*]
logic-actual[*necessitation-averse-axiom-instance*]
intro-elim-6-e **by** *blast*

13.3 Resulting Paradox

lemma *paradox*:
assumes $\bigwedge\ G\ \varphi.\ IsProperInX\ (\lambda x\ .\ (\!|G,\boldsymbol{\iota} y\ .\ \varphi\ y\ x|\!))$
shows *False*
proof –
obtain K **where** *K-def*:
$K = backdoor\ (\lambda\ x\ .\ \exists\ F\ .\ \{\!|x,F|\!\}\ \&\ \neg(\!|F,x|\!))$ **by** *auto*
have $[\exists\, x.\ (\!|A!,x^P|\!)\ \&\ (\forall\, F.\ \{\!|x^P,F|\!\} \equiv (F = K))\ in\ dw]$
using *A-objects*[*axiom-instance*] **by** *fast*
then obtain x **where** *x-prop*:
$[(\!|A!,x^P|\!)\ \&\ (\forall\, F.\ \{\!|x^P,F|\!\} \equiv (F = K))\ in\ dw]$
by (*rule Instantiate*)
{
assume $[(\!|K,x^P|\!)\ in\ dw]$
hence $[\exists\ F\ .\ \{\!|x^P,F|\!\}\ \&\ \neg(\!|F,x^P|\!)\ in\ dw]$
unfolding *K-def* **using** *fake-beta-act*[*OF assms*, *equiv-lr*]
by *blast*
then obtain F **where** *F-def*:
$[\{\!|x^P,F|\!\}\ \&\ \neg(\!|F,x^P|\!)\ in\ dw]$ **by** (*rule Instantiate*)
hence $[F = K\ in\ dw]$
using *x-prop*[*conj2*, *THEN* $\forall\, E$[**where** β=F], *equiv-lr*]
&E **unfolding** *K-def* **by** *blast*
hence $[\neg(\!|K,x^P|\!)\ in\ dw]$
using *l-identity*[*axiom-instance*,*deduction*,*deduction*]
F-def[*conj2*] **by** *fast*

```
  }
  hence 1: [¬⦇K,x^P⦈ in dw]
    using reductio-aa-1 by blast
  hence [¬(∃ F . ⦃x^P,F⦄ & ¬⦇F,x^P⦈) in dw]
    using fake-beta-act[OF assms,
          THEN oth-class-taut-5-d[equiv-lr],
          equiv-lr]
    unfolding K-def by blast
  hence [∀ F . ⦃x^P,F⦄ → ⦇F,x^P⦈ in dw]
    apply − unfolding exists-def by PLM-solver
  moreover have [⦃x^P,K⦄ in dw]
    using x-prop[conj2, THEN ∀E[where β=K], equiv-rl]
          id-eq-1 by blast
  ultimately have [⦇K,x^P⦈ in dw]
    using ∀E vdash-properties-10 by blast
  hence ⋀φ. [φ in dw]
    using raa-cor-2 1 by blast
  thus False using Semantics.T4 by auto
qed
```

13.4 Original Version of the Paradox

Originally the paradox was discovered using the following construction based on the comprehension theorem for relations without the explicit construction of the description backdoor and the resulting fake-β-conversion.

```
lemma assumes ⋀ G φ. IsProperInX (λx . ⦇G,ιy . φ y x⦈)
shows Fx-equiv-xH: [∀ H . ∃ F . □(∀x. ⦇F,x^P⦈ ≡ ⦃x^P,H⦄) in v]
proof (rule ∀I)
  fix H
  let ?G = (λx . ∀ p . p → p)
  obtain φ where φ-def: φ = (λ y x . (y^P) = x & ⦃x,H⦄) by auto
  have [∃F. □(∀x. ⦇F,x^P⦈ ≡ ⦇?G,ιy . φ y (x^P)⦈) in v]
    using relations-1[OF assms] by simp
  hence 1: [∃F. □(∀x. ⦇F,x^P⦈ ≡ (∃!y . 𝓐φ y (x^P))) in v]
    apply − apply (PLM-subst-method
        λ x . ⦇?G,ιy . φ y (x^P)⦈ λ x . (∃!y. 𝓐φ y (x^P)))
    using exe-impl-exists by auto
  then obtain F where F-def: [□(∀x. ⦇F,x^P⦈ ≡ (∃!y . 𝓐φ y (x^P))) in v]
    by (rule Instantiate)
  moreover have 2: ⋀ v x . [(∃!y . 𝓐φ y (x^P)) ≡ ⦃x^P,H⦄ in v]
  proof (rule ≡I; rule CP)
    fix x v
    assume [∃!y. 𝓐φ y (x^P) in v]
    hence [∃α. 𝓐φ α (x^P) & (∀β. 𝓐φ β (x^P) → β = α) in v]
      unfolding exists-unique-def by simp
    then obtain α where [𝓐φ α (x^P) & (∀β. 𝓐φ β (x^P) → β = α) in v]
      by (rule Instantiate)
    hence [𝓐(α^P = x^P & ⦃x^P,H⦄) in v]
      unfolding φ-def using &E by blast
    hence [𝓐(⦃x^P,H⦄) in v]
      using Act-Basic-2[equiv-lr] &E by blast
    thus [⦃x^P,H⦄ in v]
      using en-eq-10[equiv-lr] by simp
  next
    fix x v
    assume [⦃x^P,H⦄ in v]
    hence 1: [𝓐(⦃x^P,H⦄) in v]
      using en-eq-10[equiv-rl] by blast
    have [x = x in v]
      using id-eq-1[where 'a=ν] by simp
    hence [𝓐(x = x) in v]
      using id-act-3[equiv-lr] by fast
    hence [𝓐(x^P = x^P & ⦃x^P,H⦄) in v]
```

unfolding *identity-ν-def* **using** *1 Act-Basic-2*[*equiv-rl*] **&***I* **by** *blast*
hence $[\boldsymbol{\mathcal{A}}\varphi\ x\ (x^P)\ \mathit{in}\ v]$
unfolding *φ-def* **by** *simp*
moreover have $[\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta\ (x^P) \rightarrow \beta = x\ \mathit{in}\ v]$
proof (*rule* ∀*I*; *rule CP*)
fix β
assume $[\boldsymbol{\mathcal{A}}\varphi\ \beta\ (x^P)\ \mathit{in}\ v]$
hence $[\boldsymbol{\mathcal{A}}(\beta = x)\ \mathit{in}\ v]$
unfolding *φ-def identity-ν-def*
using *Act-Basic-2*[*equiv-lr*] **&***E* **by** *fast*
thus $[\beta = x\ \mathit{in}\ v]$ **using** *id-act-3*[*equiv-rl*] **by** *fast*
qed
ultimately have $[\boldsymbol{\mathcal{A}}\varphi\ x\ (x^P)\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta\ (x^P) \rightarrow \beta = x)\ \mathit{in}\ v]$
using &*I* **by** *fast*
hence $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}\varphi\ \alpha\ (x^P)\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta\ (x^P) \rightarrow \beta = \alpha)\ \mathit{in}\ v]$
by (*rule existential*)
thus $[\exists\,!y.\ \boldsymbol{\mathcal{A}}\varphi\ y\ (x^P)\ \mathit{in}\ v]$
unfolding *exists-unique-def* **by** *simp*
qed
have $[\Box(\forall\,x.\ (\!|F,x^P|\!) \equiv \{\!|x^P,H|\!\})\ \mathit{in}\ v]$
apply (*PLM-subst-goal-method*
$\lambda\varphi\ .\ \Box(\forall\,x.\ (\!|F,x^P|\!) \equiv \varphi\ x)$
$\lambda\ x\ .\ (\exists\,!y\ .\ \boldsymbol{\mathcal{A}}\varphi\ y\ (x^P)))$
using *2 F-def* **by** *auto*
thus $[\exists\ \ F\ .\ \Box(\forall\,x.\ (\!|F,x^P|\!) \equiv \{\!|x^P,H|\!\})\ \mathit{in}\ v]$
by (*rule existential*)
qed

lemma
assumes *is-propositional*: $(\bigwedge G\ \varphi.\ \mathit{IsProperInX}\ (\lambda x.\ (\!|G,\boldsymbol{\iota}y.\ \varphi\ y\ x|\!)))$
and *Abs-x*: $[(\!|A!,x^P|\!)\ \mathit{in}\ v]$
and *Abs-y*: $[(\!|A!,y^P|\!)\ \mathit{in}\ v]$
and *noteq*: $[x \neq y\ \mathit{in}\ v]$
shows *diffprop*: $[\exists\ \ F\ .\ \neg((\!|F,x^P|\!) \equiv (\!|F,y^P|\!))\ \mathit{in}\ v]$
proof −
have $[\exists\ \ F\ .\ \neg(\{\!|x^P,\ F|\!\} \equiv \{\!|y^P,\ F|\!\})\ \mathit{in}\ v]$
using *noteq* **unfolding** *exists-def*
proof (*rule reductio-aa-2*)
assume *1*: $[\forall\,F.\ \neg\neg(\{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\})\ \mathit{in}\ v]$
{
fix F
have $[(\{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\})\ \mathit{in}\ v]$
using *1*[*THEN* ∀*E*] *useful-tautologies-1*[*deduction*] **by** *blast*
}
hence $[\forall\,F.\ \{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\}\ \mathit{in}\ v]$ **by** (*rule* ∀*I*)
thus $[x = y\ \mathit{in}\ v]$
unfolding *identity-ν-def*
using *ab-obey-1*[*deduction*, *deduction*]
Abs-x Abs-y **&***I* **by** *blast*
qed
then obtain H **where** *H-def*: $[\neg(\{\!|x^P,\ H|\!\} \equiv \{\!|y^P,\ H|\!\})\ \mathit{in}\ v]$
by (*rule Instantiate*)
hence *2*: $[(\{\!|x^P,\ H|\!\}\ \&\ \neg\{\!|y^P,\ H|\!\}) \vee (\neg\{\!|x^P,\ H|\!\}\ \&\ \{\!|y^P,\ H|\!\})\ \mathit{in}\ v]$
apply − **by** *PLM-solver*
have $[\exists\,F.\ \Box(\forall\,x.\ (\!|F,x^P|\!) \equiv \{\!|x^P,H|\!\})\ \mathit{in}\ v]$
using *Fx-equiv-xH*[*OF is-propositional*, *THEN* ∀*E*] **by** *simp*
then obtain F **where** $[\Box(\forall\,x.\ (\!|F,x^P|\!) \equiv \{\!|x^P,H|\!\})\ \mathit{in}\ v]$
by (*rule Instantiate*)
hence *F-prop*: $[\forall\,x.\ (\!|F,x^P|\!) \equiv \{\!|x^P,H|\!\}\ \mathit{in}\ v]$
using *qml-2*[*axiom-instance*, *deduction*] **by** *blast*
hence *a*: $[(\!|F,x^P|\!) \equiv \{\!|x^P,H|\!\}\ \mathit{in}\ v]$
using ∀*E* **by** *blast*

```
    have b: [⦇F,y^P⦈ ≡ ⦃y^P,H⦄ in v]
      using F-prop ∀E by blast
    {
      assume 1: [⦃x^P, H⦄ & ¬⦃y^P, H⦄ in v]
      hence [⦇F,x^P⦈ in v]
        using a[equiv-rl] &E by blast
      moreover have [¬⦇F,y^P⦈ in v]
        using b[THEN oth-class-taut-5-d[equiv-lr], equiv-rl] 1[conj2] by auto
      ultimately have [⦇F,x^P⦈ & (¬⦇F,y^P⦈) in v]
        by (rule &I)
      hence [(⦇F,x^P⦈ & ¬⦇F,y^P⦈) ∨ (¬⦇F,x^P⦈ & ⦇F,y^P⦈) in v]
        using ∨I by blast
      hence [¬(⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
        using oth-class-taut-5-j[equiv-rl] by blast
    }
    moreover {
      assume 1: [¬⦃x^P, H⦄ & ⦃y^P, H⦄ in v]
      hence [⦇F,y^P⦈ in v]
        using b[equiv-rl] &E by blast
      moreover have [¬⦇F,x^P⦈ in v]
        using a[THEN oth-class-taut-5-d[equiv-lr], equiv-rl] 1[conj1] by auto
      ultimately have [¬⦇F,x^P⦈ & ⦇F,y^P⦈ in v]
        using &I by blast
      hence [(⦇F,x^P⦈ & ¬⦇F,y^P⦈) ∨ (¬⦇F,x^P⦈ & ⦇F,y^P⦈) in v]
        using ∨I by blast
      hence [¬(⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
        using oth-class-taut-5-j[equiv-rl] by blast
    }
    ultimately have [¬(⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
      using 2 intro-elim-4-b reductio-aa-1 by blast
    thus [∃ F . ¬(⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
      by (rule existential)
  qed

  lemma original-paradox:
    assumes is-propositional: (⋀G φ. IsProperInX (λx. ⦇G,ιy. φ y x⦈))
    shows False
  proof -
    fix v
    have [∃ x y. ⦇A!,x^P⦈ & ⦇A!,y^P⦈ & x ≠ y & (∀ F. ⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
      using aclassical2 by auto
    then obtain x where
      [∃ y. ⦇A!,x^P⦈ & ⦇A!,y^P⦈ & x ≠ y & (∀ F. ⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
      by (rule Instantiate)
    then obtain y where xy-def:
      [⦇A!,x^P⦈ & ⦇A!,y^P⦈ & x ≠ y & (∀ F. ⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
      by (rule Instantiate)
    have [∃ F . ¬(⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
      using diffprop[OF assms, OF xy-def[conj1,conj1,conj1],
                    OF xy-def[conj1,conj1,conj2],
                    OF xy-def[conj1,conj2]]
      by auto
    then obtain F where [¬(⦇F,x^P⦈ ≡ ⦇F,y^P⦈) in v]
      by (rule Instantiate)
    moreover have [⦇F,x^P⦈ ≡ ⦇F,y^P⦈ in v]
      using xy-def[conj2] by (rule ∀E)
    ultimately have ⋀φ.[φ in v]
      using PLM.raa-cor-2 by blast
    thus False
      using Semantics.T4 by auto
  qed

end
```