Embedding of the Theory of Abstract Objects in Isabelle/HOL

Daniel Kirchner

March 3, 2017

Abstract

This document constitutes a core contribution of the MSc project of Daniel Kirchner. The supervisor of this project is Christoph Benzmller. The project idea results from an ongoing collaboration between Benzmller 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
typedef o = $UNIV::(j \Rightarrow i \Rightarrow bool)$ set
morphisms $evalo$ $makeo$ **..** — truth values

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

type-synonym $\Pi_0 = o$ — zero place relations
typedef Π_1 = $UNIV::(\upsilon \Rightarrow j \Rightarrow i \Rightarrow bool)$ set
morphisms $eval\Pi_1$ $make\Pi_1$ **..** — one place relations
typedef Π_2 = $UNIV::(\upsilon \Rightarrow \upsilon \Rightarrow j \Rightarrow i \Rightarrow bool)$ set
morphisms $eval\Pi_2$ $make\Pi_2$ **..** — two place relations
typedef Π_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

Remark 1. *Individual terms can be definite descriptions which may not denote. The condition under which an individual term denotes is stored as a boolean. Note that relation terms on the other hand* *always denote, so there is no need for a distinction between relation terms and relation variables.*

typedef κ = *UNIV*::(*bool*×ν) *set* **morphisms** *eval*κ *make*κ **..**

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

Remark 2. *Individual terms can be explicitely marked to represent only denoting resp. logically proper objects.*

lift-definition $\nu\kappa$ :: $\nu \Rightarrow \kappa$ (-^{P} [*90*] *90*) **is** *Pair True* **.**
lift-definition *denotes* :: $\kappa \Rightarrow$*bool* **is** *fst* **.**
lift-definition *denotation* :: $\kappa \Rightarrow \nu$ **is** *snd* **.**

1.2 Mapping from abstract objects to special Urelements

consts $\alpha\sigma$ :: $\alpha \Rightarrow \sigma$
axiomatization where $\alpha\sigma$-*surj*: *surj* $\alpha\sigma$

1.3 Conversion between objects and Urelements

definition $\nu\upsilon$ :: $\nu \Rightarrow \upsilon$ **where** $\nu\upsilon \equiv$ *case-*ν $\omega\upsilon$ ($\sigma\upsilon \circ \alpha\sigma$)
definition $\upsilon\nu$ :: $\upsilon \Rightarrow \nu$ **where** $\upsilon\nu \equiv$ *case-*υ $\omega\nu$ ($\alpha\nu \circ$ (*inv* $\alpha\sigma$))

1.4 Exemplification of n-place relations.

Remark 3. *An exemplification formula is only true if all individual variables denote. Furthermore exemplification only depends on the Urelement corresponding to the individual.*

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

1.5 Encoding

Remark 4. *An encoding formula is again only true if the individual term denotes. Furthermore ordinary objects never encode, whereas abstract objects encode a property if and only if the property is contained in it as per the Aczel Model.*

lift-definition *enc* :: $\kappa \Rightarrow \Pi_1 \Rightarrow$o (⦃-,-⦄) **is**
 λ *x F w s* . (*denotes x*) $\wedge$ *case-*ν (λ ω . *False*) (λ α . *F* $\in \alpha$) (*denotation x*) **.**

1.6 Connectives and Quantifiers

Remark 5. *The connectives behave classically if evaluated for the actual state dj, whereas their behavior is governed by uninterpreted constants for any other state.*

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

lift-definition $not :: \mathrm{o} \Rightarrow \mathrm{o}$ ($\neg$- $[54]$ 70) **is**
$\lambda\ p\ s\ w\ .\ s = dj \wedge \neg p\ dj\ w \vee s \neq dj \wedge (I\text{-}NOT\ s\ (p\ s)\ w)$ **.**
lift-definition $impl :: \mathrm{o} \Rightarrow \mathrm{o} \Rightarrow \mathrm{o}$ (**infixl** $\rightarrow$ 51) **is**
$\lambda\ p\ q\ s\ w\ .\ s = dj \wedge (p\ dj\ w \longrightarrow q\ dj\ w) \vee s \neq dj \wedge (I\text{-}IMPL\ s\ (p\ s)\ (q\ s))\ w$ **.**
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**
$\lambda\ p\ s\ w\ .\ p\ dj\ dw$ **.**

1.7 Definite Description

Remark 6. *Definite descriptions map conditions on individual variables to individual terms. Whether the condition is satisfied by a unique individual (and therefore the definite description denotes) is stored as a boolean.*

lift-definition $that::(\nu \Rightarrow \mathrm{o}) \Rightarrow \kappa$ (**binder** $\boldsymbol{\iota}$ $[8]$ 9) **is**
$\lambda\ \varphi\ .\ (\exists!\ x\ .\ (\varphi\ x)\ dj\ dw,\ THE\ x\ .\ (\varphi\ x)\ dj\ dw)$ **.**

1.8 Lambda Expressions

Remark 7. *Lambda expressions map functions acting on individual variables to functions acting on Urelements (i.e. relations). Note that the inverse mapping $\upsilon\nu$ is injective only for ordinary objects. As propositional formulas, which are the only terms PM allows inside lambda expressions, do not contain encoding subformulas, they only depends on Urelements, though. For propositional formulas the lambda expressions therefore exactly correspond to the lambda expressions in PM. Lambda expressions with non-propositional formulas, which are not allowed in PM, because in general they lead to inconsistencies, have a non-standard semantics.* $\boldsymbol{\lambda}\ x\ .\ \{\!|x^P, F|\!\}$ *can* *be translated to "being x such that there exists an abstract object, which encodes F, that is mapped to the same Urelement as x" instead of "being x such that x encodes F". This construction avoids the aforementioned inconsistencies.*

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

1.9 Validity

Remark 8. *A formula is considered semantically valid for a possible world, if it evaluates to True for the actual state and the given possible world.*

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

1.10 Concreteness

Remark 9. *In order to define concreteness, 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 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.*

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

abbreviation *OrdinaryObjectsPossiblyConcrete* **where**
OrdinaryObjectsPossiblyConcrete $\equiv \forall\ x\ .\ \exists\ v\ .$ *ConcreteInWorld x v*
abbreviation *PossiblyContingentObjectExists* **where**
PossiblyContingentObjectExists $\equiv \exists\ x\ v\ .$ *ConcreteInWorld x v*
$\wedge$ ($\exists\ w\ .\ \neg$ *ConcreteInWorld x w*)
abbreviation *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. *Concreteness of ordinary objects can now be defined using this axiomatized uninterpreted constant. Abstract objects on the other hand are never concrete.*

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

1.11 Automation

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*]]

1.12 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 $\upsilon\nu$-$\omega\upsilon$-*is*-$\omega\nu$[*meta-aux*]: $\upsilon\nu$ ($\omega\upsilon$ *x*) = $\omega\nu$ *x* **by** (*simp add*: $\upsilon\nu$-*def*)
lemma *denotation-proper*[*meta-aux*]: *denotation* (x^P) = *x*
by (*simp add*: *meta-aux* $\nu\kappa$-*def denotation-def*)
lemma *proper-denotes*[*meta-aux*]: *denotes* (x^P)
by (*simp add*: *meta-aux* $\nu\kappa$-*def denotes-def*)
lemma *proper-denotation*[*meta-aux*]: *denotation* (x^P) = *x*
by (*simp add*: *meta-aux* $\nu\kappa$-*def denotation-def*)
lemma $\nu\upsilon$-$\upsilon\nu$-*id*[*meta-aux*]: $\nu\upsilon$ ($\upsilon\nu$ (*x*)) = *x*
by (*simp add*: $\nu\upsilon$-*def* $\upsilon\nu$-*def* $\alpha\sigma$-*surj surj-f-inv-f split*: υ.*split*)
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** $\nu\upsilon$-$\upsilon\nu$-*id surjI* **by** *blast*
lemma $\upsilon\nu\kappa$-*aux1*[*meta-aux*]:
fst (*eval*κ ($\upsilon\nu$ ($\nu\upsilon$ (*snd* (*eval*κ *x*)))^P))
apply *transfer*
by *simp*
lemma $\upsilon\nu\kappa$-*aux2*[*meta-aux*]:
($\nu\upsilon$ (*snd* (*eval*κ ($\upsilon\nu$ ($\nu\upsilon$ (*snd* (*eval*κ *x*)))^P)))) = ($\nu\upsilon$ (*snd* (*eval*κ *x*)))
apply *transfer*
using $\nu\upsilon$-$\upsilon\nu$-*id* **by** *auto*

2 Basic Definitions

2.1 Derived Connectives

definition *diamond*::o⇒o (◇- [*62*] *63*) **where**
  diamond ≡ λ φ . ¬□¬φ
definition *conj*::o⇒o⇒o (**infixl &** *53*) **where**
  conj ≡ λ x y . ¬(x → ¬y)
definition *disj*::o⇒o⇒o (**infixl** ∨ *52*) **where**
  disj ≡ λ x y . ¬x → y
definition *equiv*::o⇒o⇒o (**infixl** ≡ *51*) **where**
  equiv ≡ λ x y . (x → y) **&** (y → x)

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

2.2 Abstract and Ordinary Objects

definition *Ordinary* :: Π_1 (*O*!) **where** *Ordinary* ≡ **λ**x. ◇⦇E!,x^P⦈
definition *Abstract* :: Π_1 (*A*!) **where** *Abstract* ≡ **λ**x. ¬◇⦇E!,x^P⦈

2.3 Identity Definitions

definition *basic-identity*$_E$::Π_2 **where**
  basic-identity$_E$ ≡ $\boldsymbol{\lambda}^2$ (λ x y . ⦇O!,x^P⦈ **&** ⦇O!,y^P⦈
    & □($\forall_1$ F. ⦇F,x^P⦈ ≡ ⦇F,y^P⦈))

definition *basic-identity*$_E$*-infix*::κ⇒κ⇒o (**infixl** $=_E$ *63*) **where**
  x $=_E$ y ≡ ⦇*basic-identity*$_E$, x, y⦈

definition *basic-identity*$_\kappa$ (**infixl** $=_\kappa$ *63*) **where**
  basic-identity$_\kappa$ ≡ λ x y . (x $=_E$ y) ∨ ⦇A!,x⦈ **&** ⦇A!,y⦈
    & □($\forall_1$ F. ⦃x,F⦄ ≡ ⦃y,F⦄)

definition *basic-identity*$_1$ (**infixl** $=_1$ *63*) **where**
  basic-identity$_1$ ≡ λ F G . □($\forall_\nu$ x. ⦃x^P,F⦄ ≡ ⦃x^P,G⦄)

definition *basic-identity*$_2$ :: Π_2⇒Π_2⇒o (**infixl** $=_2$ *63*) **where**
  basic-identity$_2$ ≡ λ F G . $\forall_\nu$ x. ((**λ**y. ⦇F,x^P,y^P⦈) $=_1$ (**λ**y. ⦇G,x^P,y^P⦈))
    & ((**λ**y. ⦇F,y^P,x^P⦈) $=_1$ (**λ**y. ⦇G,y^P,x^P⦈))

definition *basic-identity*$_3$::Π_3⇒Π_3⇒o (**infixl** $=_3$ *63*) **where**
  basic-identity$_3$ ≡ λ F G . $\forall_\nu$ x y. (**λ**z. ⦇F,z^P,x^P,y^P⦈) $=_1$ (**λ**z. ⦇G,z^P,x^P,y^P⦈)
    & (**λ**z. ⦇F,x^P,z^P,y^P⦈) $=_1$ (**λ**z. ⦇G,x^P,z^P,y^P⦈)
    & (**λ**z. ⦇F,x^P,y^P,z^P⦈) $=_1$ (**λ**z. ⦇G,x^P,y^P,z^P⦈)

definition *basic-identity*$_o$::o⇒o⇒o (**infixl** $=_o$ *63*) **where**
  basic-identity$_o$ ≡ λ F G . (**λ**y. F) $=_1$ (**λ**y. G)

3 Semantics

3.1 Propositional Formulas

Remark 11. *The embedding extends the notion of propositional formulas to functions that are propositional in the individual variables that are their parameters, i.e. their parameters only occur in exemplification and not in encoding subformulas. This weaker condition is enough to prove the semantics of propositional formulas.*

named-theorems *IsPropositional-intros*

definition *IsPropositionalInX* :: (κ⇒o)⇒*bool* **where**
IsPropositionalInX ≡ λ Θ . ∃ χ . Θ = (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*⦈))

lemma *IsPropositionalInX-intro*[*IsPropositional-intros*]:
IsPropositionalInX (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 *IsPropositionalInX-def* **by** *blast*

definition *IsPropositionalInXY* :: (κ⇒κ⇒o)⇒*bool* **where**
IsPropositionalInXY ≡ λ Θ . ∃ χ . Θ = (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*⦈)

$(\lambda\ F\ a\ .\ (|F,a,x,y|))$
$(*$ *three place* (y,x) $*)$ $(\lambda\ F\ a\ .\ (|F,y,x,a|))$ $(\lambda\ F\ a\ .\ (|F,y,a,x|))$
$(\lambda\ F\ a\ .\ (|F,a,y,x|))$
$(*$ *three place* (x,x,y) $*)$ $(\lambda\ F\ .\ (|F,x,x,y|))$ $(\lambda\ F\ .\ (|F,x,y,x|))$ $(\lambda\ F\ .\ (|F,y,x,x|))$
$(*$ *three place* (x,y,y) $*)$ $(\lambda\ F\ .\ (|F,x,y,y|))$ $(\lambda\ F\ .\ (|F,y,x,y|))$ $(\lambda\ F\ .\ (|F,y,y,x|))$
$(*$ *three place* (x,x,x) $*)$ $(\lambda\ F\ .\ (|F,x,x,x|))$
$(*$ *three place* (y,y,y) $*)$ $(\lambda\ F\ .\ (|F,y,y,y|)))$

lemma *IsPropositionalInXY-intro*[*IsPropositional-intros*]:
IsPropositionalInXY $(\lambda\ x\ y\ .\ \chi$
$(*$ *only* x $*)$
$(*$ *one place* $*)$ $(\lambda\ F\ .\ (|F,x|))$
$(*$ *two place* $*)$ $(\lambda\ F\ .\ (|F,x,x|))$ $(\lambda\ F\ a\ .\ (|F,x,a|))$ $(\lambda\ F\ a\ .\ (|F,a,x|))$
$(*$ *three place three* x $*)$ $(\lambda\ F\ .\ (|F,x,x,x|))$
$(*$ *three place two* x $*)$ $(\lambda\ F\ a\ .\ (|F,x,x,a|))$ $(\lambda\ F\ a\ .\ (|F,x,a,x|))$
$(\lambda\ F\ a\ .\ (|F,a,x,x|))$
$(*$ *three place one* x $*)$ $(\lambda\ F\ a\ b.\ (|F,x,a,b|))$ $(\lambda\ F\ a\ b.\ (|F,a,x,b|))$
$(\lambda\ F\ a\ b\ .\ (|F,a,b,x|))$
$(*$ *only* y $*)$
$(*$ *one place* $*)$ $(\lambda\ F\ .\ (|F,y|))$
$(*$ *two place* $*)$ $(\lambda\ F\ .\ (|F,y,y|))$ $(\lambda\ F\ a\ .\ (|F,y,a|))$ $(\lambda\ F\ a\ .\ (|F,a,y|))$
$(*$ *three place three* y $*)$ $(\lambda\ F\ .\ (|F,y,y,y|))$
$(*$ *three place two* y $*)$ $(\lambda\ F\ a\ .\ (|F,y,y,a|))$ $(\lambda\ F\ a\ .\ (|F,y,a,y|))$
$(\lambda\ F\ a\ .\ (|F,a,y,y|))$
$(*$ *three place one* y $*)$ $(\lambda\ F\ a\ b.\ (|F,y,a,b|))$ $(\lambda\ F\ a\ b.\ (|F,a,y,b|))$
$(\lambda\ F\ a\ b\ .\ (|F,a,b,y|))$
$(*$ x *and* y $*)$
$(*$ *two place* $*)$ $(\lambda\ F\ .\ (|F,x,y|))$ $(\lambda\ F\ .\ (|F,y,x|))$
$(*$ *three place* (x,y) $*)$ $(\lambda\ F\ a\ .\ (|F,x,y,a|))$ $(\lambda\ F\ a\ .\ (|F,x,a,y|))$
$(\lambda\ F\ a\ .\ (|F,a,x,y|))$
$(*$ *three place* (y,x) $*)$ $(\lambda\ F\ a\ .\ (|F,y,x,a|))$ $(\lambda\ F\ a\ .\ (|F,y,a,x|))$
$(\lambda\ F\ a\ .\ (|F,a,y,x|))$
$(*$ *three place* (x,x,y) $*)$ $(\lambda\ F\ .\ (|F,x,x,y|))$ $(\lambda\ F\ .\ (|F,x,y,x|))$
$(\lambda\ F\ .\ (|F,y,x,x|))$
$(*$ *three place* (x,y,y) $*)$ $(\lambda\ F\ .\ (|F,x,y,y|))$ $(\lambda\ F\ .\ (|F,y,x,y|))$
$(\lambda\ F\ .\ (|F,y,y,x|))$
$(*$ *three place* (x,x,x) $*)$ $(\lambda\ F\ .\ (|F,x,x,x|))$
$(*$ *three place* (y,y,y) $*)$ $(\lambda\ F\ .\ (|F,y,y,y|)))$
unfolding *IsPropositionalInXY-def* **by** *metis*

definition *IsPropositionalInXYZ* :: $(\kappa\Rightarrow\kappa\Rightarrow\kappa\Rightarrow o)\Rightarrow$*bool* **where**
IsPropositionalInXYZ $\equiv \lambda\ \Theta\ .\ \exists\ \chi\ .\ \Theta = (\lambda\ x\ y\ z\ .\ \chi$
$(*$ *only* x $*)$
$(*$ *one place* $*)$ $(\lambda\ F\ .\ (|F,x|))$
$(*$ *two place* $*)$ $(\lambda\ F\ .\ (|F,x,x|))$ $(\lambda\ F\ a\ .\ (|F,x,a|))$ $(\lambda\ F\ a\ .\ (|F,a,x|))$
$(*$ *three place three* x $*)$ $(\lambda\ F\ .\ (|F,x,x,x|))$
$(*$ *three place two* x $*)$ $(\lambda\ F\ a\ .\ (|F,x,x,a|))$ $(\lambda\ F\ a\ .\ (|F,x,a,x|))$
$(\lambda\ F\ a\ .\ (|F,a,x,x|))$
$(*$ *three place one* x $*)$ $(\lambda\ F\ a\ b.\ (|F,x,a,b|))$ $(\lambda\ F\ a\ b.\ (|F,a,x,b|))$
$(\lambda\ F\ a\ b\ .\ (|F,a,b,x|))$
$(*$ *only* y $*)$
$(*$ *one place* $*)$ $(\lambda\ F\ .\ (|F,y|))$

$(*\ \textit{two place}\ *)\ (\lambda\ F\ .\ (\!|F,y,y|\!))\ (\lambda\ F\ a\ .\ (\!|F,y,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,a,y|\!))$
$(*\ \textit{three place three}\ y\ *)\ (\lambda\ F\ .\ (\!|F,y,y,y|\!))$
$(*\ \textit{three place two}\ y\ *)\ (\lambda\ F\ a\ .\ (\!|F,y,y,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,y,a,y|\!))$
$(\lambda\ F\ a\ .\ (\!|F,a,y,y|\!))$
$(*\ \textit{three place one}\ y\ *)\ (\lambda\ F\ a\ b.\ (\!|F,y,a,b|\!))\ (\lambda\ F\ a\ b.\ (\!|F,a,y,b|\!))$
$(\lambda\ F\ a\ b\ .\ (\!|F,a,b,y|\!))$
$(*\ \textit{only}\ z\ *)$
$(*\ \textit{one place}\ *)\ (\lambda\ F\ .\ (\!|F,z|\!))$
$(*\ \textit{two place}\ *)\ (\lambda\ F\ .\ (\!|F,z,z|\!))\ (\lambda\ F\ a\ .\ (\!|F,z,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,a,z|\!))$
$(*\ \textit{three place three}\ z\ *)\ (\lambda\ F\ .\ (\!|F,z,z,z|\!))$
$(*\ \textit{three place two}\ z\ *)\ (\lambda\ F\ a\ .\ (\!|F,z,z,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,z,a,z|\!))$
$(\lambda\ F\ a\ .\ (\!|F,a,z,z|\!))$
$(*\ \textit{three place one}\ z\ *)\ (\lambda\ F\ a\ b.\ (\!|F,z,a,b|\!))\ (\lambda\ F\ a\ b.\ (\!|F,a,z,b|\!))$
$(\lambda\ F\ a\ b\ .\ (\!|F,a,b,z|\!))$
$(*\ x\ \textit{and}\ y\ *)$
$(*\ \textit{two place}\ *)\ (\lambda\ F\ .\ (\!|F,x,y|\!))\ (\lambda\ F\ .\ (\!|F,y,x|\!))$
$(*\ \textit{three place}\ (x,y)\ *)\ (\lambda\ F\ a\ .\ (\!|F,x,y,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,x,a,y|\!))$
$(\lambda\ F\ a\ .\ (\!|F,a,x,y|\!))$
$(*\ \textit{three place}\ (y,x)\ *)\ (\lambda\ F\ a\ .\ (\!|F,y,x,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,y,a,x|\!))$
$(\lambda\ F\ a\ .\ (\!|F,a,y,x|\!))$
$(*\ \textit{three place}\ (x,x,y)\ *)\ (\lambda\ F\ .\ (\!|F,x,x,y|\!))\ (\lambda\ F\ .\ (\!|F,x,y,x|\!))$
$(\lambda\ F\ .\ (\!|F,y,x,x|\!))$
$(*\ \textit{three place}\ (x,y,y)\ *)\ (\lambda\ F\ .\ (\!|F,x,y,y|\!))\ (\lambda\ F\ .\ (\!|F,y,x,y|\!))$
$(\lambda\ F\ .\ (\!|F,y,y,x|\!))$
$(*\ \textit{three place}\ (x,x,x)\ *)\ (\lambda\ F\ .\ (\!|F,x,x,x|\!))$
$(*\ \textit{three place}\ (y,y,y)\ *)\ (\lambda\ F\ .\ (\!|F,y,y,y|\!))$
$(*\ x\ \textit{and}\ z\ *)$
$(*\ \textit{two place}\ *)\ (\lambda\ F\ .\ (\!|F,x,z|\!))\ (\lambda\ F\ .\ (\!|F,z,x|\!))$
$(*\ \textit{three place}\ (x,z)\ *)\ (\lambda\ F\ a\ .\ (\!|F,x,z,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,x,a,z|\!))$
$(\lambda\ F\ a\ .\ (\!|F,a,x,z|\!))$
$(*\ \textit{three place}\ (z,x)\ *)\ (\lambda\ F\ a\ .\ (\!|F,z,x,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,z,a,x|\!))$
$(\lambda\ F\ a\ .\ (\!|F,a,z,x|\!))$
$(*\ \textit{three place}\ (x,x,z)\ *)\ (\lambda\ F\ .\ (\!|F,x,x,z|\!))\ (\lambda\ F\ .\ (\!|F,x,z,x|\!))$
$(\lambda\ F\ .\ (\!|F,z,x,x|\!))$
$(*\ \textit{three place}\ (x,z,z)\ *)\ (\lambda\ F\ .\ (\!|F,x,z,z|\!))\ (\lambda\ F\ .\ (\!|F,z,x,z|\!))$
$(\lambda\ F\ .\ (\!|F,z,z,x|\!))$
$(*\ \textit{three place}\ (x,x,x)\ *)\ (\lambda\ F\ .\ (\!|F,x,x,x|\!))$
$(*\ \textit{three place}\ (z,z,z)\ *)\ (\lambda\ F\ .\ (\!|F,z,z,z|\!))$
$(*\ y\ \textit{and}\ z\ *)$
$(*\ \textit{two place}\ *)\ (\lambda\ F\ .\ (\!|F,y,z|\!))\ (\lambda\ F\ .\ (\!|F,z,y|\!))$
$(*\ \textit{three place}\ (y,z)\ *)\ (\lambda\ F\ a\ .\ (\!|F,y,z,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,y,a,z|\!))$
$(\lambda\ F\ a\ .\ (\!|F,a,y,z|\!))$
$(*\ \textit{three place}\ (z,y)\ *)\ (\lambda\ F\ a\ .\ (\!|F,z,y,a|\!))\ (\lambda\ F\ a\ .\ (\!|F,z,a,y|\!))$
$(\lambda\ F\ a\ .\ (\!|F,a,z,y|\!))$
$(*\ \textit{three place}\ (y,y,z)\ *)\ (\lambda\ F\ .\ (\!|F,y,y,z|\!))\ (\lambda\ F\ .\ (\!|F,y,z,y|\!))$
$(\lambda\ F\ .\ (\!|F,z,y,y|\!))$
$(*\ \textit{three place}\ (y,z,z)\ *)\ (\lambda\ F\ .\ (\!|F,y,z,z|\!))\ (\lambda\ F\ .\ (\!|F,z,y,z|\!))$
$(\lambda\ F\ .\ (\!|F,z,z,y|\!))$
$(*\ \textit{three place}\ (y,y,y)\ *)\ (\lambda\ F\ .\ (\!|F,y,y,y|\!))$
$(*\ \textit{three place}\ (z,z,z)\ *)\ (\lambda\ F\ .\ (\!|F,z,z,z|\!))$
$(*\ x\ y\ z\ *)$
$(*\ \textit{three place}\ (x,...)\ *)\ (\lambda\ F\ .\ (\!|F,x,y,z|\!))\ (\lambda\ F\ .\ (\!|F,x,z,y|\!))$
$(*\ \textit{three place}\ (y,...)\ *)\ (\lambda\ F\ .\ (\!|F,y,x,z|\!))\ (\lambda\ F\ .\ (\!|F,y,z,x|\!))$
$(*\ \textit{three place}\ (z,...)\ *)\ (\lambda\ F\ .\ (\!|F,z,x,y|\!))\ (\lambda\ F\ .\ (\!|F,z,y,x|\!)))$

lemma *IsPropositionalInXYZ-intro*[*IsPropositional-intros*]:
IsPropositionalInXYZ $(\lambda\ x\ y\ z\ .\ \chi$
(∗ *only x* ∗)
(∗ *one place* ∗) $(\lambda\ F\ .\ (|F,x|))$
(∗ *two place* ∗) $(\lambda\ F\ .\ (|F,x,x|))$ $(\lambda\ F\ a\ .\ (|F,x,a|))$ $(\lambda\ F\ a\ .\ (|F,a,x|))$
(∗ *three place three x* ∗) $(\lambda\ F\ .\ (|F,x,x,x|))$
(∗ *three place two x* ∗) $(\lambda\ F\ a\ .\ (|F,x,x,a|))$ $(\lambda\ F\ a\ .\ (|F,x,a,x|))$
$(\lambda\ F\ a\ .\ (|F,a,x,x|))$
(∗ *three place one x* ∗) $(\lambda\ F\ a\ b.\ (|F,x,a,b|))$ $(\lambda\ F\ a\ b.\ (|F,a,x,b|))$
$(\lambda\ F\ a\ b\ .\ (|F,a,b,x|))$
(∗ *only y* ∗)
(∗ *one place* ∗) $(\lambda\ F\ .\ (|F,y|))$
(∗ *two place* ∗) $(\lambda\ F\ .\ (|F,y,y|))$ $(\lambda\ F\ a\ .\ (|F,y,a|))$ $(\lambda\ F\ a\ .\ (|F,a,y|))$
(∗ *three place three y* ∗) $(\lambda\ F\ .\ (|F,y,y,y|))$
(∗ *three place two y* ∗) $(\lambda\ F\ a\ .\ (|F,y,y,a|))$ $(\lambda\ F\ a\ .\ (|F,y,a,y|))$
$(\lambda\ F\ a\ .\ (|F,a,y,y|))$
(∗ *three place one y* ∗) $(\lambda\ F\ a\ b.\ (|F,y,a,b|))$ $(\lambda\ F\ a\ b.\ (|F,a,y,b|))$
$(\lambda\ F\ a\ b\ .\ (|F,a,b,y|))$
(∗ *only z* ∗)
(∗ *one place* ∗) $(\lambda\ F\ .\ (|F,z|))$
(∗ *two place* ∗) $(\lambda\ F\ .\ (|F,z,z|))$ $(\lambda\ F\ a\ .\ (|F,z,a|))$ $(\lambda\ F\ a\ .\ (|F,a,z|))$
(∗ *three place three z* ∗) $(\lambda\ F\ .\ (|F,z,z,z|))$
(∗ *three place two z* ∗) $(\lambda\ F\ a\ .\ (|F,z,z,a|))$ $(\lambda\ F\ a\ .\ (|F,z,a,z|))$
$(\lambda\ F\ a\ .\ (|F,a,z,z|))$
(∗ *three place one z* ∗) $(\lambda\ F\ a\ b.\ (|F,z,a,b|))$ $(\lambda\ F\ a\ b.\ (|F,a,z,b|))$
$(\lambda\ F\ a\ b\ .\ (|F,a,b,z|))$
(∗ *x and y* ∗)
(∗ *two place* ∗) $(\lambda\ F\ .\ (|F,x,y|))$ $(\lambda\ F\ .\ (|F,y,x|))$
(∗ *three place* (x,y) ∗) $(\lambda\ F\ a\ .\ (|F,x,y,a|))$ $(\lambda\ F\ a\ .\ (|F,x,a,y|))$
$(\lambda\ F\ a\ .\ (|F,a,x,y|))$
(∗ *three place* (y,x) ∗) $(\lambda\ F\ a\ .\ (|F,y,x,a|))$ $(\lambda\ F\ a\ .\ (|F,y,a,x|))$
$(\lambda\ F\ a\ .\ (|F,a,y,x|))$
(∗ *three place* (x,x,y) ∗) $(\lambda\ F\ .\ (|F,x,x,y|))$ $(\lambda\ F\ .\ (|F,x,y,x|))$
$(\lambda\ F\ .\ (|F,y,x,x|))$
(∗ *three place* (x,y,y) ∗) $(\lambda\ F\ .\ (|F,x,y,y|))$ $(\lambda\ F\ .\ (|F,y,x,y|))$
$(\lambda\ F\ .\ (|F,y,y,x|))$
(∗ *three place* (x,x,x) ∗) $(\lambda\ F\ .\ (|F,x,x,x|))$
(∗ *three place* (y,y,y) ∗) $(\lambda\ F\ .\ (|F,y,y,y|))$
(∗ *x and z* ∗)
(∗ *two place* ∗) $(\lambda\ F\ .\ (|F,x,z|))$ $(\lambda\ F\ .\ (|F,z,x|))$
(∗ *three place* (x,z) ∗) $(\lambda\ F\ a\ .\ (|F,x,z,a|))$ $(\lambda\ F\ a\ .\ (|F,x,a,z|))$
$(\lambda\ F\ a\ .\ (|F,a,x,z|))$
(∗ *three place* (z,x) ∗) $(\lambda\ F\ a\ .\ (|F,z,x,a|))$ $(\lambda\ F\ a\ .\ (|F,z,a,x|))$
$(\lambda\ F\ a\ .\ (|F,a,z,x|))$
(∗ *three place* (x,x,z) ∗) $(\lambda\ F\ .\ (|F,x,x,z|))$ $(\lambda\ F\ .\ (|F,x,z,x|))$
$(\lambda\ F\ .\ (|F,z,x,x|))$
(∗ *three place* (x,z,z) ∗) $(\lambda\ F\ .\ (|F,x,z,z|))$ $(\lambda\ F\ .\ (|F,z,x,z|))$
$(\lambda\ F\ .\ (|F,z,z,x|))$
(∗ *three place* (x,x,x) ∗) $(\lambda\ F\ .\ (|F,x,x,x|))$
(∗ *three place* (z,z,z) ∗) $(\lambda\ F\ .\ (|F,z,z,z|))$
(∗ *y and z* ∗)
(∗ *two place* ∗) $(\lambda\ F\ .\ (|F,y,z|))$ $(\lambda\ F\ .\ (|F,z,y|))$
(∗ *three place* (y,z) ∗) $(\lambda\ F\ a\ .\ (|F,y,z,a|))$ $(\lambda\ F\ a\ .\ (|F,y,a,z|))$

$(\lambda\ F\ a\ .\ (|F,a,y,z|))$
$(*\ \textit{three place}\ (z,y)\ *)\ (\lambda\ F\ a\ .\ (|F,z,y,a|))\ (\lambda\ F\ a\ .\ (|F,z,a,y|))$
$(\lambda\ F\ a\ .\ (|F,a,z,y|))$
$(*\ \textit{three place}\ (y,y,z)\ *)\ (\lambda\ F\ .\ (|F,y,y,z|))\ (\lambda\ F\ .\ (|F,y,z,y|))$
$(\lambda\ F\ .\ (|F,z,y,y|))$
$(*\ \textit{three place}\ (y,z,z)\ *)\ (\lambda\ F\ .\ (|F,y,z,z|))\ (\lambda\ F\ .\ (|F,z,y,z|))$
$(\lambda\ F\ .\ (|F,z,z,y|))$
$(*\ \textit{three place}\ (y,y,y)\ *)\ (\lambda\ F\ .\ (|F,y,y,y|))$
$(*\ \textit{three place}\ (z,z,z)\ *)\ (\lambda\ F\ .\ (|F,z,z,z|))$
$(*\ x\ y\ z\ *)$
$(*\ \textit{three place}\ (x,...)\ *)\ (\lambda\ F\ .\ (|F,x,y,z|))\ (\lambda\ F\ .\ (|F,x,z,y|))$
$(*\ \textit{three place}\ (y,...)\ *)\ (\lambda\ F\ .\ (|F,y,x,z|))\ (\lambda\ F\ .\ (|F,y,z,x|))$
$(*\ \textit{three place}\ (z,...)\ *)\ (\lambda\ F\ .\ (|F,z,x,y|))\ (\lambda\ F\ .\ (|F,z,y,x|)))$
unfolding *IsPropositionalInXYZ-def* **by** *metis*

named-theorems *IsPropositionalIn-defs*
declare *IsPropositionalInX-def* [*IsPropositionalIn-defs*]
IsPropositionalInXY-def [*IsPropositionalIn-defs*]
IsPropositionalInXYZ-def [*IsPropositionalIn-defs*]

3.2 Semantics

locale *Semantics*
begin
named-theorems *semantics*

The domains for the terms in the language.

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 $W = i$

Denotations of the terms in the language.

lift-definition $d_\kappa :: \kappa \Rightarrow R_\kappa\ option$ **is**
$\lambda\ x\ .\ (\textit{if fst x then Some}\ (\textit{snd}\ x)\ \textit{else None})$ **.**
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* **.**

Designated actual world.

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

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* $\equiv \lambda\ F\ w\ .\ \{\ (x,y,z)\ .\ F\ (\nu\upsilon\ x)\ (\nu\upsilon\ y)\ (\nu\upsilon\ z)\ dj\ w\ \}$

Encoding extensions.

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

Collect 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*]

Semantics for exemplification and encoding.

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*
by (*simp add*: *meta-defs meta-aux denotation-def denotes-def*)
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*
by (*simp add*: *meta-defs meta-aux denotation-def denotes-def*)
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*
by (*simp add*: *meta-defs meta-aux denotation-def denotes-def*)

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*
by (*simp add*: *meta-defs meta-aux denotation-def denotes-def split*: ν.*split*)

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*)

Semantics for connectives and quantifiers.

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*)

Semantics for descriptions and lambda expressions.

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

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*]: $d_3\ (\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ (\!|F,\ x^P,\ y^P,\ z^P|\!))) = d_3\ F$
by (*simp add*: *meta-defs meta-aux*)

lemma *D5-1*[*semantics*]:
assumes *IsPropositionalInX* φ
shows $\bigwedge\ w\ o_1\ r\ .\ \mathit{Some}\ r = d_1\ (\boldsymbol{\lambda}\ x\ .\ (\varphi\ (x^P))) \land \mathit{Some}\ o_1 = d_\kappa\ x$
$\longrightarrow (o_1 \in \mathit{ex1}\ r\ w) = (w \models \varphi\ x)$
using *assms* **unfolding** *IsPropositionalIn-defs semantics-defs*
by (*auto simp*: *meta-defs meta-aux denotes-def denotation-def*)

lemma *D5-2*[*semantics*]:
assumes *IsPropositionalInXY* φ
shows $\bigwedge\ w\ o_1\ o_2\ r\ .\ \mathit{Some}\ r = d_2\ (\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \varphi\ (x^P)\ (y^P)))$
$\land\ \mathit{Some}\ o_1 = d_\kappa\ x \land \mathit{Some}\ o_2 = d_\kappa\ y$
$\longrightarrow ((o_1,o_2) \in \mathit{ex2}\ r\ w) = (w \models \varphi\ x\ y)$

using *assms* **unfolding** *IsPropositionalIn-defs semantics-defs*
by (*auto simp*: *meta-defs meta-aux denotes-def denotation-def*)

lemma *D5-3*[*semantics*]:
assumes *IsPropositionalInXYZ* φ
shows $\bigwedge$ $w\ o_1\ o_2\ o_3\ r$. *Some* $r = d_3$ ($\boldsymbol{\lambda}^3$ ($\lambda\ x\ y\ z$. φ (x^P) (y^P) (z^P)))
$\wedge$ *Some* $o_1 = d_\kappa\ x \wedge$ *Some* $o_2 = d_\kappa\ y \wedge$ *Some* $o_3 = d_\kappa\ z$
$\longrightarrow$ ((o_1,o_2,o_3) $\in$ *ex3* $r\ w$) = ($w \models \varphi\ x\ y\ z$)
using *assms* **unfolding** *IsPropositionalIn-defs semantics-defs*
by (*auto simp*: *meta-defs meta-aux denotes-def denotation-def*)

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

Auxiliary lemmata.

lemma *propex*$_1$: $\exists$ r . *Some* $r = d_1$ F
unfolding d_1*-def* **by** *simp*
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_κ*-inject*: $\bigwedge x\ y\ o_1$. *Some* $o_1 = d_\kappa\ x \wedge$ *Some* $o_1 = d_\kappa\ y \Longrightarrow x = y$
proof −
fix $x :: \kappa$ **and** $y :: \kappa$ **and** $o_1 :: \nu$
assume *Some* $o_1 = d_\kappa\ x \wedge$ *Some* $o_1 = d_\kappa\ y$
moreover hence
fst (*eval*κ x) $\wedge$ *fst* (*eval*κ y) $\wedge$ *snd* (*eval*κ x) = o_1 $\wedge$ *snd* (*eval*κ x) = o_1
unfolding d_κ*-def*
apply *transfer*
apply *simp*
by (*metis option.distinct*(*1*) *option.inject*)
ultimately show $x = y$
unfolding d_κ*-def*
apply *transfer*
by *auto*
qed
lemma d_κ*-proper*: d_κ (u^P) = *Some* u
unfolding d_κ*-def* **by** (*simp add*: $\nu\kappa$*-def meta-aux*)
end

3.3 Validity Syntax

abbreviation *validity-in* :: o⇒*i*⇒*bool* ([- *in* -] [*1*]) **where**
validity-in ≡ $\lambda\ \varphi\ v$. $v \models \varphi$
abbreviation *actual-validity* :: o⇒*bool* ([-] [*1*]) **where**
actual-validity ≡ $\lambda\ \varphi$. $dw \models \varphi$
abbreviation *necessary-validity* :: o⇒*bool* (□[-] [*1*]) **where**
necessary-validity ≡ $\lambda\ \varphi$. $\forall\ v$. ($v \models \varphi$)

4 MetaSolver

Remark 12. *meta-solver is a resolution prover that translates expressions in the embedded logic to expressions in the meta-logic as far as possible. The rules for connectives and quantifiers are simple, whereas the rules for exemplification and encoding are more verbose. Futhermore rules for the defined identities are proven. By design the defined identities in the embedded logic coincides 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*))
)+

4.1 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*)

4.2 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*)

4.3 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*)

4.4 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*]: $[\varphi \equiv \psi \text{ in } v] \Longrightarrow ([\varphi \text{ in } v] \longleftrightarrow [\psi \text{ in } v])$
by (*auto simp*: *equiv-def NotS ImplS ConjS*)
lemma *EquivS*[*meta-subst*]: $[\varphi \equiv \psi \text{ in } v] = ([\varphi \text{ in } v] \longleftrightarrow [\psi \text{ in } v])$
by (*auto simp*: *equiv-def NotS ImplS ConjS*)

4.5 Rules for Disjunction

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

4.6 Rules for Necessity

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

4.7 Rules for Possibility

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

4.8 Rules for Quantification

lemma $All_\nu I$[*meta-intro*]: $(\bigwedge x{::}\nu.\ [\varphi\ x \text{ in } v]) \Longrightarrow [\forall_\nu\ x.\ \varphi\ x \text{ in } v]$
by (*auto simp*: *Semantics.T8-*ν)
lemma $All_\nu E$[*meta-elim*]: $[\forall_\nu x.\ \varphi\ x \text{ in } v] \Longrightarrow (\bigwedge x{::}\nu.[\varphi\ x \text{ in } v])$
by (*auto simp*: *Semantics.T8-*ν)
lemma $All_\nu S$[*meta-subst*]: $[\forall_\nu x.\ \varphi\ x \text{ in } v] = (\forall\, x{::}\nu.[\varphi\ x \text{ in } v])$
by (*auto simp*: *Semantics.T8-*ν)

lemma $All_0 I$[*meta-intro*]: $(\bigwedge x{::}\Pi_0.\ [\varphi\ x \text{ in } v]) \Longrightarrow [\forall_0\ x.\ \varphi\ x \text{ in } v]$
by (*auto simp*: *Semantics.T8-0*)
lemma $All_0 E$[*meta-elim*]: $[\forall_0\ x.\ \varphi\ x \text{ in } v] \Longrightarrow (\bigwedge x{::}\Pi_0\ .[\varphi\ x \text{ in } v])$
by (*auto simp*: *Semantics.T8-0*)
lemma $All_0 S$[*meta-subst*]: $[\forall_0\ x.\ \varphi\ x \text{ in } v] = (\forall\, x{::}\Pi_0.[\varphi\ x \text{ in } v])$
by (*auto simp*: *Semantics.T8-0*)

lemma $All_1 I$[*meta-intro*]: $(\bigwedge x{::}\Pi_1.\ [\varphi\ x \text{ in } v]) \Longrightarrow [\forall_1\ x.\ \varphi\ x \text{ in } v]$
by (*auto simp*: *Semantics.T8-1*)
lemma $All_1 E$[*meta-elim*]: $[\forall_1\ x.\ \varphi\ x \text{ in } v] \Longrightarrow (\bigwedge x{::}\Pi_1\ .[\varphi\ x \text{ in } v])$
by (*auto simp*: *Semantics.T8-1*)
lemma $All_1 S$[*meta-subst*]: $[\forall_1\ x.\ \varphi\ x \text{ in } v] = (\forall\, x{::}\Pi_1.[\varphi\ x \text{ in } v])$

by (*auto simp*: *Semantics.T8-1*)

lemma All_2I[*meta-intro*]: $(\bigwedge x{::}\Pi_2.\ [\varphi\ x\ in\ v]) \Longrightarrow [\forall_2\ x.\ \varphi\ x\ in\ v]$
by (*auto simp*: *Semantics.T8-2*)
lemma All_2E[*meta-elim*]: $[\forall_2\ x.\ \varphi\ x\ in\ v] \Longrightarrow (\bigwedge x{::}\Pi_2\ .[\varphi\ x\ in\ v])$
by (*auto simp*: *Semantics.T8-2*)
lemma All_2S[*meta-subst*]: $[\forall_2\ x.\ \varphi\ x\ in\ v] = (\forall x{::}\Pi_2.[\varphi\ x\ in\ v])$
by (*auto simp*: *Semantics.T8-2*)

lemma All_3I[*meta-intro*]: $(\bigwedge x{::}\Pi_3.\ [\varphi\ x\ in\ v]) \Longrightarrow [\forall_3\ x.\ \varphi\ x\ in\ v]$
by (*auto simp*: *Semantics.T8-3*)
lemma All_3E[*meta-elim*]: $[\forall_3\ x.\ \varphi\ x\ in\ v] \Longrightarrow (\bigwedge x{::}\Pi_3\ .[\varphi\ x\ in\ v])$
by (*auto simp*: *Semantics.T8-3*)
lemma All_3S[*meta-subst*]: $[\forall_3\ x.\ \varphi\ x\ in\ v] = (\forall x{::}\Pi_3.[\varphi\ x\ in\ v])$
by (*auto simp*: *Semantics.T8-3*)

4.9 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*)

4.10 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*)

4.11 Rules for Exemplification

4.11.1 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*)

4.11.2 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*)

4.11.3 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*]:
$[(\!|F,x,y|\!)\ in\ v] = (\exists\ r\ o_1\ o_2\ .\ Some\ r = d_2\ F \wedge Some\ o_1 = d_\kappa\ x$
$\wedge\ Some\ o_2 = d_\kappa\ y \wedge (o_1,\ o_2) \in ex2\ r\ v)$
by (*auto simp*: *Semantics.T1-2*)

4.11.4 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 $[(\!|F,x,y,z|\!)\ in\ v]$
using *assms* **by** (*auto simp*: *Semantics.T1-3*)
lemma *Exe3E*[*meta-elim*]:
assumes $[(\!|F,x,y,z|\!)\ in\ v]$
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*]:
$[(\!|F,x,y,z|\!)\ in\ v] = (\exists\ r\ o_1\ o_2\ o_3\ .\ Some\ r = d_3\ F \wedge Some\ o_1 = d_\kappa\ x$
$\wedge\ Some\ o_2 = d_\kappa\ y \wedge Some\ o_3 = d_\kappa\ z$
$\wedge\ (o_1,\ o_2,\ o_3) \in ex3\ r\ v)$
by (*auto simp*: *Semantics.T1-3*)

4.12 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 $[(\!|O!,x|\!)\ in\ v]$
proof $-$
obtain o_1 **and** y **where** *1*: $Some\ o_1 = d_\kappa\ x \wedge o_1 = \omega\nu\ y$
using *assms* **by** *auto*
moreover obtain v **where** *ConcreteInWorld y v*
using *OrdinaryObjectsPossiblyConcreteAxiom* **by** *auto*
ultimately show *?thesis*
unfolding *Ordinary-def conn-defs meta-defs*
apply (*simp add*: *meta-aux*)
apply *transfer*
by (*metis* (*full-types*) $\nu\upsilon$-$\omega\nu$-*is*-$\omega\upsilon$ υ*.simps*(*5*)
option.distinct(*1*) *option.sel*)
qed
lemma *OrdE*[*meta-elim*]:
assumes $[(\!|O!,x|\!)\ in\ v]$
shows $\exists\ o_1\ y.\ Some\ o_1 = d_\kappa\ x \wedge o_1 = \omega\nu\ y$
using *assms* **unfolding** *Ordinary-def conn-defs meta-defs*
apply (*simp add*: *meta-aux* d_κ-*def denotes-def denotation-def*)
by (*metis* ν*.exhaust* ν*.simps*(*6*) $\nu\upsilon$-*def* υ*.simps*(*6*) *comp-apply*)
lemma *OrdS*[*meta-cong*]:
$[(\!|O!,x|\!)\ in\ v] = (\exists\ o_1\ y.\ Some\ o_1 = d_\kappa\ x \wedge o_1 = \omega\nu\ y)$
using *OrdI OrdE* **by** *blast*

4.13 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 $[(\!|A!,x|\!)\ in\ v]$
proof $-$
obtain o_1 y **where** $Some\ o_1 = d_\kappa\ x \wedge o_1 = \alpha\nu\ y$
using *assms* **by** *auto*
thus *?thesis*
unfolding *Abstract-def conn-defs meta-defs*
apply (*simp add*: *meta-aux*)
by (*metis* d_κ-*inject* d_κ-*proper* ν*.simps*(*6*) $\nu\upsilon$-*def* υ*.simps*(*6*)
o-apply proper-denotation proper-denotes)
qed
lemma *AbsE*[*meta-elim*]:
assumes $[(\!|A!,x|\!)\ in\ v]$
shows $\exists\ o_1\ y.\ Some\ o_1 = d_\kappa\ x \wedge o_1 = \alpha\nu\ y$
using *assms* **unfolding** *conn-defs meta-defs Abstract-def*
apply (*simp add*: *meta-aux* d_κ-*def denotes-def denotation-def*)
by (*metis OrdinaryObjectsPossiblyConcreteAxiom* ν*.exhaust*
$\nu\upsilon$-$\omega\nu$-*is*-$\omega\upsilon$ υ*.simps*(*5*))
lemma *AbsS*[*meta-cong*]:
$[(\!|A!,x|\!)\ in\ v] = (\exists\ o_1\ y.\ Some\ o_1 = d_\kappa\ x \wedge o_1 = \alpha\nu\ y)$
using *AbsI AbsE* **by** *blast*

4.14 Rules for Definite Descriptions

lemma *TheS*: $(\iota x.\ \varphi\ x) = make\kappa\ (\exists!\ x\ .\ evalo\ (\varphi\ x)\ dj\ dw,$
$THE\ x\ .\ evalo\ (\varphi\ x)\ dj\ dw)$
by (*auto simp*: *meta-defs*)

4.15 Rules for Identity

4.15.1 Ordinary Objects

lemma $Eq_E I$[*meta-intro*]:
assumes $\exists\ o_1\ X\ o_2.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = o_2 \wedge o_1 = \omega\nu\ X$
shows $[x =_E y\ in\ v]$
using *assms*
apply (*simp add*: *meta-defs meta-aux basic-identity$_E$-def basic-identity$_E$-infix-def conn-defs Ordinary-def OrdinaryObjectsPossiblyConcreteAxiom denotes-def Semantics.d_κ-def split*: ν.*split* υ.*split*)
using *OrdinaryObjectsPossiblyConcreteAxiom*
apply *transfer*
apply *simp*
by (*metis* $\nu\upsilon$-$\omega\nu$-*is*-$\omega\upsilon$ υ.*distinct*(*1*) υ.*inject*(*1*) *option.distinct*(*1*) *option.sel*)
lemma $Eq_E E$[*meta-elim*]:
assumes $[x =_E y\ in\ v]$
shows $\exists\ o_1\ X\ o_2.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = o_2 \wedge o_1 = \omega\nu\ X$
proof −
have *1*: $[(\!|O!,x|\!)\ \&\ (\!|O!,y|\!)\ \&\ \Box(\forall_1\ F.\ (\!|F,x|\!) \equiv (\!|F,y|\!))\ in\ v]$
using *assms* **unfolding** *basic-identity$_E$-def basic-identity$_E$-infix-def*
using *D4-2 T1-2 D5-2 IsPropositional-intros* **by** *meson*
hence *2*: $\exists\ o_1\ o_2\ X\ Y\ .\ Some\ o_1 = d_\kappa\ x \wedge o_1 = \omega\nu\ X \wedge Some\ o_2 = d_\kappa\ y \wedge o_2 = \omega\nu\ Y$
apply (*subst* (*asm*) *ConjS*)
apply (*subst* (*asm*) *ConjS*)
using *OrdE* **by** *auto*
then obtain $o_1\ o_2\ X\ Y$ **where** *3*:
$Some\ o_1 = d_\kappa\ x \wedge o_1 = \omega\nu\ X \wedge Some\ o_2 = d_\kappa\ y \wedge o_2 = \omega\nu\ Y$
by *auto*
have $\exists\ r\ .\ Some\ r = d_1\ (\boldsymbol{\lambda}\ z\ .\ make\mathrm{o}\ (\lambda\ w\ s\ .\ d_\kappa\ (z^P) = Some\ o_1))$
using $propex_1$ **by** *auto*
then obtain r **where** *4*:
$Some\ r = d_1\ (\boldsymbol{\lambda}\ z\ .\ make\mathrm{o}\ (\lambda\ w\ s\ .\ d_\kappa\ (z^P) = Some\ o_1))$
by *auto*
hence *5*: $r = (\lambda u\ w\ s.\ Some\ (\upsilon\nu\ u) = Some\ o_1)$
unfolding *lambdabinder1-def d_1-def d_κ-proper*
apply *transfer*
by *simp*
have $[\Box(\forall_1\ F.\ (\!|F,x|\!) \equiv (\!|F,y|\!))\ in\ v]$
using *1* **using** *ConjE* **by** *blast*
hence *6*: $\forall\ v\ F\ .\ [(\!|F,x|\!)\ in\ v] \longleftrightarrow [(\!|F,y|\!)\ in\ v]$
using *BoxE EquivE All$_1$E* **by** *fast*
hence *7*: $\forall\ v\ .\ (o_1 \in ex1\ r\ v) = (o_2 \in ex1\ r\ v)$
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 proper-denotation proper-denotes valid-in.abs-eq*)
have $o_1 \in ex1\ r\ v$
using *5 3* **unfolding** *ex1-def* **by** (*simp add*: *meta-aux*)

hence $o_2 \in ex1\ r\ v$
  using *7* **by** *auto*
hence $o_1 = o_2$
  unfolding *ex1-def 5* **using** *3* **by** (*auto simp*: *meta-aux*)
thus *?thesis*
  using *3* **by** *auto*
qed — TODO: simplify this
lemma $Eq_E S$[*meta-subst*]:
$[x =_E\ y\ in\ v] = (\exists\ o_1\ X\ o_2.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = o_2 \wedge o_1 = \omega\nu\ X)$
using $Eq_E I$ $Eq_E E$ **by** *blast*

4.15.2 Individuals

lemma $Eq\kappa I$[*meta-intro*]:
assumes $\exists\ o_1\ o_2.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = o_2$
shows $[x =_\kappa\ y\ in\ v]$
proof –
have $x = y$ **using** *assms* d_κ*-inject* **by** *meson*
moreover have $[x =_\kappa\ x\ in\ v]$
  unfolding *basic-identity*$_\kappa$*-def*
  apply *meta-solver*
  by (*metis* (*no-types*, *lifting*) *assms AbsI Exe1E* ν*.exhaust*)
ultimately show *?thesis* **by** *auto*
qed
lemma $Eq\kappa$*-prop*:
assumes $[x =_\kappa\ y\ in\ v]$
shows $[\varphi\ x\ in\ v] = [\varphi\ y\ in\ v]$
proof –
have $[x =_E\ y \vee (|A!,x|)\ \&\ (|A!,y|)\ \&\ \Box(\forall_1\ F.\ \{|x,F|\} \equiv \{|y,F|\})\ in\ v]$
  using *assms* **unfolding** *basic-identity*$_\kappa$*-def* **by** *simp*
moreover {
  assume $[x =_E\ y\ in\ v]$
  hence $(\exists\ o_1\ o_2.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = o_2)$
    using $Eq_E E$ **by** *fast*
}
moreover {
  assume *1*: $[(|A!,x|)\ \&\ (|A!,y|)\ \&\ \Box(\forall_1\ F.\ \{|x,F|\} \equiv \{|y,F|\})\ in\ v]$
  hence *2*: $(\exists\ o_1\ o_2\ X\ Y.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = \alpha\nu\ X \wedge o_2 = \alpha\nu\ Y)$
    using *AbsE ConjE* **by** *meson*
  moreover then obtain $o_1\ o_2\ X\ Y$ **where** *3*:
    $Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = \alpha\nu\ X \wedge o_2 = \alpha\nu\ Y$
    by *auto*
  moreover have *4*: $[\Box(\forall_1\ F.\ \{|x,F|\} \equiv \{|y,F|\})\ in\ v]$
    using *1 ConjE* **by** *blast*
  hence *6*: $\forall\ v\ F\ .\ [\{|x,F|\}\ in\ v] \longleftrightarrow [\{|y,F|\}\ in\ v]$
    using *BoxE* $All_1 E$ *EquivE* **by** *fast*
  hence *7*: $\forall v\ r.\ (\exists\ o_1.\ Some\ o_1 = d_\kappa\ x \wedge o_1 \in en\ r) = (\exists\ o_1.\ Some\ o_1 = d_\kappa\ y \wedge o_1 \in en\ r)$
    apply *cut-tac* **apply** *meta-solver*
    using $propex_1$ d_1*-inject* **apply** *simp*
    apply *transfer* **by** *simp*
  hence *8*: $\forall\ r.\ (o_1 \in en\ r) = (o_2 \in en\ r)$

using *3* d_κ*-inject* d_κ*-proper* **apply** *simp*
by (*metis option.inject*)
hence $\forall r.\ (o_1 \in r) = (o_2 \in r)$
unfolding *en-def* **using** *3*
by (*metis Collect-cong Collect-mem-eq* ν*.simps*(*6*)
mem-Collect-eq makeΠ_1*-cases*)
hence $(o_1 \in \{\ x\ .\ o_1 = x\ \}) = (o_2 \in \{\ x\ .\ o_1 = x\ \})$
by *metis*
hence $o_1 = o_2$ **by** *simp*
hence $(\exists\ o_1\ o_2.\ Some\ o_1 = d_\kappa\ x \wedge Some\ o_2 = d_\kappa\ y \wedge o_1 = o_2)$
using *3* **by** *auto*
}
ultimately have $x = y$
using *DisjS* **using** *Semantics.*d_κ*-inject* **by** *auto*
thus $(v \models (\varphi\ x)) = (v \models (\varphi\ y))$ **by** *simp*
qed
lemma *Eq*κ*E*[*meta-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 Eq*κ*-prop* **by** *blast*
moreover obtain φ **where** φ*-prop*:
$\varphi = (\lambda\ \alpha\ .\ make\mathrm{o}\ (\lambda\ w\ s\ .\ (\exists\ o_1\ o_2.\ Some\ o_1 = d_\kappa\ x$
$\wedge\ Some\ o_2 = d_\kappa\ \alpha \wedge o_1 = o_2)))$
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*κ*S*[*meta-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 *Eq*κ*I Eq*κ*E* **by** *blast*

4.15.3 One-Place Relations

lemma *Eq*$_1$*I*[*meta-intro*]: $F = G \Longrightarrow [F =_1 G\ in\ v]$
unfolding *basic-identity*$_1$*-def*
apply (*rule BoxI*, *rule All*$_\nu$*I*, *rule EquivI*)
by *simp*
lemma *Eq*$_1$*E*[*meta-elim*]: $[F =_1 G\ in\ v] \Longrightarrow F = G$
unfolding *basic-identity*$_1$*-def*
apply (*drule BoxE*, *drule-tac x*=($\alpha\nu\ \{\ F\ \}$) **in** *All*$_\nu$*E*, *drule EquivE*)
apply (*simp add*: *Semantics.T2*)
unfolding *en-def* d_κ*-def* d_1*-def*
using *proper-denotation proper-denotes*
by (*simp add*: *denotation-def denotes-def meta-aux*)

lemma $Eq_1S[meta\text{-}subst]$: $[F =_1 G \ in \ v] = (F = G)$
using Eq_1I Eq_1E **by** *auto*
lemma $Eq_1\text{-}prop$: $[F =_1 G \ in \ v] \Longrightarrow [\varphi \ F \ in \ v] = [\varphi \ G \ in \ v]$
using Eq_1E **by** *blast*

4.15.4 Two-Place Relations

lemma $Eq_2I[meta\text{-}intro]$: $F = G \Longrightarrow [F =_2 G \ in \ v]$
unfolding *basic-identity*$_2$*-def*
apply (*rule* $All_\nu I$, *rule ConjI*, (*subst* Eq_1S)+)
by *simp*
lemma $Eq_2E[meta\text{-}elim]$: $[F =_2 G \ in \ v] \Longrightarrow F = G$
proof −
assume $[F =_2 G \ in \ v]$
hence $[\forall_\nu \ x. \ (\boldsymbol{\lambda} y. \ (|F,x^P,y^P|)) =_1 (\boldsymbol{\lambda} y. \ (|G,x^P,y^P|)) \ in \ v]$
unfolding *basic-identity*$_2$*-def*
apply *cut-tac* **apply** *meta-solver* **by** *auto*
hence $\bigwedge x. \ (make\Pi_1 \ (eval\Pi_2 \ F \ (\nu\upsilon \ x)) = make\Pi_1 \ ((eval\Pi_2 \ G \ (\nu\upsilon \ x))))$
apply *cut-tac* **apply** *meta-solver*
by (*simp add*: *meta-defs meta-aux*)
hence $\bigwedge x. \ (eval\Pi_2 \ F \ (\nu\upsilon \ x) = eval\Pi_2 \ G \ (\nu\upsilon \ x))$
by (*simp add*: $make\Pi_1$*-inject*)
hence $\bigwedge x1. \ (eval\Pi_2 \ F \ x1) = (eval\Pi_2 \ G \ x1)$
using $\nu\upsilon$*-surj* **by** (*metis* $\nu\upsilon$-$\upsilon\nu$*-id*)
thus $F = G$ **using** $eval\Pi_2$*-inject* **by** *blast*
qed
lemma $Eq_2S[meta\text{-}subst]$: $[F =_2 G \ in \ v] = (F = G)$
using Eq_2I Eq_2E **by** *auto*
lemma $Eq_2\text{-}prop$: $[F =_2 G \ in \ v] \Longrightarrow [\varphi \ F \ in \ v] = [\varphi \ G \ in \ v]$
using Eq_2E **by** *blast*

4.15.5 Three-Place Relations

lemma $Eq_3I[meta\text{-}intro]$: $F = G \Longrightarrow [F =_3 G \ in \ v]$
apply (*simp add*: *meta-defs meta-aux conn-defs basic-identity*$_3$*-def*)
using $MetaSolver.Eq_1I$ *valid-in.rep-eq* **by** *auto*
lemma $Eq_3E[meta\text{-}elim]$: $[F =_3 G \ in \ v] \Longrightarrow F = G$
proof −
assume $[F =_3 G \ in \ v]$
hence $[\forall_\nu \ x \ y. \ (\boldsymbol{\lambda} z. \ (|F,x^P,y^P,z^P|)) =_1 (\boldsymbol{\lambda} z. \ (|G,x^P,y^P,z^P|)) \ in \ v]$
unfolding *basic-identity*$_3$*-def* **apply** *cut-tac*
apply *meta-solver* **by** *auto*
hence $\bigwedge x \ y. \ (\boldsymbol{\lambda} z. \ (|F,x^P,y^P,z^P|)) = (\boldsymbol{\lambda} z. \ (|G,x^P,y^P,z^P|))$
using Eq_1E $All_\nu S$ **by** (*metis* (*mono-tags*, *lifting*))
hence $\bigwedge x \ y. \ make\Pi_1 \ (eval\Pi_3 \ F \ x \ y) = make\Pi_1 \ (eval\Pi_3 \ G \ x \ y)$
apply (*auto simp*: *meta-defs meta-aux*)
using $\nu\upsilon$*-surj* **by** (*metis* $\nu\upsilon$-$\upsilon\nu$*-id*)
thus $F = G$ **using** $make\Pi_1$*-inject* $eval\Pi_3$*-inject* **by** *blast*
qed
lemma $Eq_3S[meta\text{-}subst]$: $[F =_3 G \ in \ v] = (F = G)$
using Eq_3I Eq_3E **by** *auto*
lemma $Eq_3\text{-}prop$: $[F =_3 G \ in \ v] \Longrightarrow [\varphi \ F \ in \ v] = [\varphi \ G \ in \ v]$
using Eq_3E **by** *blast*

4.15.6 Propositions

lemma $Eq_o I$[*meta-intro*]: $x = y \Longrightarrow [x =_o y \text{ in } v]$
  unfolding *basic-identity*$_o$*-def* **by** (*simp add*: $Eq_1 S$)
lemma $Eq_o E$[*meta-elim*]: $[F =_o G \text{ in } v] \Longrightarrow F = G$
  unfolding *basic-identity*$_o$*-def*
  apply (*drule* $Eq_1 E$)
  apply (*simp add*: *meta-defs*)
  using *evalo-inject make*Π_1*-inject*
  by (*metis UNIV-I*)
lemma $Eq_o S$[*meta-subst*]: $[F =_o G \text{ in } v] = (F = G)$
  using $Eq_o I$ $Eq_o E$ **by** *auto*
lemma Eq_o*-prop*: $[F =_o G \text{ in } v] \Longrightarrow [\varphi\ F \text{ in } v] = [\varphi\ G \text{ in } v]$
  using $Eq_o E$ **by** *blast*

end

5 General Quantification

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

5.1 Type Class

Datatype for types for which quantification is defined:

datatype $var = \nu var\ (var\nu\colon \nu) \mid o var\ (var o\colon o) \mid \Pi_1 var\ (var\Pi_1\colon \Pi_1)$
  $\mid \Pi_2 var\ (var\Pi_2\colon \Pi_2) \mid \Pi_3 var\ (var\Pi_3\colon \Pi_3)$

Type class for quantifiable types:

class *quantifiable* = **fixes** *forall* :: $('a \Rightarrow o) \Rightarrow o$ (**binder** $\forall$ [*8*] *9*)
  and *qvar* :: $'a \Rightarrow var$
  and *varq* :: $var \Rightarrow 'a$
  assumes *quantifiable-T8*: $(w \models (\forall\ x\ .\ \psi\ x)) = (\forall\ x\ .\ (w \models (\psi\ x)))$
    and *varq-qvar-id*: *varq* (*qvar* x) = x
begin
  definition *exists* :: $('a \Rightarrow o) \Rightarrow o$ (**binder** $\exists$ [*8*] *9*) **where**
    $exists \equiv \lambda\ \varphi\ .\ \neg(\forall\ x\ .\ \neg\varphi\ x)$
  declare *exists-def*[*conn-defs*]
end

Semantics for the general all quantifier:

lemma (**in** *Semantics*) *T8*: **shows** $(w \models \forall\ x\ .\ \psi\ x) = (\forall\ x\ .\ (w \models \psi\ x))$
  using *quantifiable-T8* **.**

5.2 Instantiations

instantiation ν :: *quantifiable*

```
begin
  definition forall-ν :: (ν⇒o)⇒o where forall-ν ≡ forall_ν
  definition qvar-ν :: ν⇒var where qvar ≡ νvar
  definition varq-ν :: var⇒ν where varq ≡ varν
  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 .
  next
    fix x :: ν
    show varq (qvar x) = x
      unfolding qvar-ν-def varq-ν-def by simp
  qed
end

instantiation o :: quantifiable
begin
  definition forall-o :: (o⇒o)⇒o where forall-o ≡ forall_o
  definition qvar-o :: o⇒var where qvar ≡ ovar
  definition varq-o :: var⇒o where varq ≡ varo
  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 .
  next
    fix x :: o
    show varq (qvar x) = x
      unfolding qvar-o-def varq-o-def by simp
  qed
end

instantiation Π1 :: quantifiable
begin
  definition forall-Π1 :: (Π1⇒o)⇒o where forall-Π1 ≡ forall1
  definition qvar-Π1 :: Π1⇒var where qvar ≡ Π1var
  definition varq-Π1 :: var⇒Π1 where varq ≡ varΠ1
  instance proof
    fix w :: i and ψ :: Π1⇒o
    show (w ⊨ ∀ x. ψ x) = (∀ x. (w ⊨ ψ x))
      unfolding forall-Π1-def using Semantics.T8-1 .
  next
    fix x :: Π1
    show varq (qvar x) = x
      unfolding qvar-Π1-def varq-Π1-def by simp
  qed
end

instantiation Π2 :: quantifiable
begin
  definition forall-Π2 :: (Π2⇒o)⇒o where forall-Π2 ≡ forall2
  definition qvar-Π2 :: Π2⇒var where qvar ≡ Π2var
  definition varq-Π2 :: var⇒Π2 where varq ≡ varΠ2
  instance proof
    fix w :: i and ψ :: Π2⇒o
```

show $(w \models \forall x.\ \psi\ x) = (\forall x.\ (w \models \psi\ x))$
unfolding *forall*-Π_2-*def* **using** *Semantics.T8-2* **.**
next
fix $x :: \Pi_2$
show *varq* (*qvar* x) $= x$
unfolding *qvar*-Π_2-*def* *varq*-Π_2-*def* **by** *simp*
qed
end

instantiation Π_3 :: *quantifiable*
begin
definition *forall*-Π_3 :: $(\Pi_3 \Rightarrow \mathrm{o}) \Rightarrow \mathrm{o}$ **where** *forall*-$\Pi_3 \equiv$ *forall*$_3$
definition *qvar*-Π_3 :: $\Pi_3 \Rightarrow$ *var* **where** *qvar* $\equiv \Pi_3$*var*
definition *varq*-Π_3 :: *var* $\Rightarrow \Pi_3$ **where** *varq* $\equiv$ *var*Π_3
instance proof
fix $w :: i$ **and** $\psi :: \Pi_3 \Rightarrow \mathrm{o}$
show $(w \models \forall x.\ \psi\ x) = (\forall x.\ (w \models \psi\ x))$
unfolding *forall*-Π_3-*def* **using** *Semantics.T8-3* **.**
next
fix $x :: \Pi_3$
show *varq* (*qvar* x) $= x$
unfolding *qvar*-Π_3-*def* *varq*-Π_3-*def* **by** *simp*
qed
end

5.3 MetaSolver Rules

Remark 14. *The meta-solver is extended by rules for general quantification.*

context *MetaSolver*
begin

5.3.1 Rules for General All Quantification.

lemma *AllI*[*meta-intro*]: $(\bigwedge x::'a::\mathit{quantifiable}.\ [\varphi\ x\ \mathit{in}\ v]) \Longrightarrow [\forall\ x.\ \varphi\ x\ \mathit{in}\ v]$
by (*auto simp*: *Semantics.T8*)
lemma *AllE*[*meta-elim*]: $[\forall x.\ \varphi\ x\ \mathit{in}\ v] \Longrightarrow (\bigwedge x::'a::\mathit{quantifiable}.[\varphi\ x\ \mathit{in}\ v])$
by (*auto simp*: *Semantics.T8*)
lemma *AllS*[*meta-subst*]: $[\forall x.\ \varphi\ x\ \mathit{in}\ v] = (\forall x::'a::\mathit{quantifiable}.[\varphi\ x\ \mathit{in}\ v])$
by (*auto simp*: *Semantics.T8*)

5.3.2 Rules for Existence

lemma *ExIRule*: $([\varphi\ y\ \mathit{in}\ v]) \Longrightarrow [\exists x.\ \varphi\ x\ \mathit{in}\ v]$
by (*auto simp*: *exists-def NotS AllS*)
lemma *ExI*[*meta-intro*]: $(\exists\ y\ .\ [\varphi\ y\ \mathit{in}\ v]) \Longrightarrow [\exists x.\ \varphi\ x\ \mathit{in}\ v]$
by (*auto simp*: *exists-def NotS AllS*)
lemma *ExE*[*meta-elim*]: $[\exists x.\ \varphi\ x\ \mathit{in}\ v] \Longrightarrow (\exists\ y\ .\ [\varphi\ y\ \mathit{in}\ v])$
by (*auto simp*: *exists-def NotS AllS*)
lemma *ExS*[*meta-subst*]: $[\exists x.\ \varphi\ x\ \mathit{in}\ v] = (\exists\ y\ .\ [\varphi\ y\ \mathit{in}\ v])$

by (*auto simp*: *exists-def NotS AllS*)
lemma *ExERule*: **assumes** $[\exists\, x.\ \varphi\ x\ in\ v]$ **obtains** *x* **where** $[\varphi\ x\ in\ v]$

using *ExE assms* **by** *auto*

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 of equality which is needed to state the axioms later. This type class is then instantiated for all applicable types.*

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** ≠ *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** ∃! [*8*] *9*) **where**
exists-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* ≡ *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 $(v \models \varphi\ (denotation\ (\alpha^P))) \Longrightarrow (v \models \varphi\ (denotation\ (\beta^P)))$
by *meson*
thus $(v \models \varphi\ \alpha) \Longrightarrow (v \models \varphi\ \beta)$
by (*simp only*: *proper-denotation*)
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*$_o$
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*$_o$-*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 used the type specific quantifiers and identities. We now introduce equivalent alternative definitions that use the general identity and general quantifiers.*

named-theorems *identity-defs*
lemma *identity$_E$-def* [*identity-defs*]:
  basic-identity$_E$ $\equiv \boldsymbol{\lambda}^2$ $(\lambda x\ y.\ (\!|O!,x^P|\!)$ **&** $(\!|O!,y^P|\!)$ **&** $\Box(\forall F.\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!)))$
  unfolding *basic-identity$_E$-def forall-Π_1-def* **by** *simp*
lemma *identity$_E$-infix-def* [*identity-defs*]:
  $x =_E y \equiv (\!|$*basic-identity$_E$*$,x,y|\!)$ **using** *basic-identity$_E$-infix-def* **.**
lemma *identity$_\kappa$-def* [*identity-defs*]:
  $op = \equiv \lambda x\ y.\ x =_E y \vee (\!|A!,x|\!)$ **&** $(\!|A!,y|\!)$ **&** $\Box(\forall\ F.\ \{\!|x,F|\!\} \equiv \{\!|y,F|\!\})$
  unfolding *identity-κ-def basic-identity$_\kappa$-def forall-Π_1-def* **by** *simp*
lemma *identity$_\nu$-def* [*identity-defs*]:
  $op = \equiv \lambda x\ y.\ (x^P) =_E (y^P) \vee (\!|A!,x^P|\!)$ **&** $(\!|A!,y^P|\!)$ **&** $\Box(\forall\ F.\ \{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\})$
  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-identity$_1$-def forall-ν-def* **by** *simp*
lemma *identity$_2$-def* [*identity-defs*]:
  $op = \equiv \lambda F\ G.\ \forall\ x.\ (\boldsymbol{\lambda}y.\ (\!|F,x^P,y^P|\!)) = (\boldsymbol{\lambda}y.\ (\!|G,x^P,y^P|\!))$
        & $(\boldsymbol{\lambda}y.\ (\!|F,y^P,x^P|\!)) = (\boldsymbol{\lambda}y.\ (\!|G,y^P,x^P|\!))$
  unfolding *identity-Π_2-def identity-Π_1-def basic-identity$_2$-def forall-ν-def* **by** *simp*
lemma *identity$_3$-def* [*identity-defs*]:
  $op = \equiv \lambda F\ G.\ \forall\ x\ y.\ (\boldsymbol{\lambda}z.\ (\!|F,z^P,x^P,y^P|\!)) = (\boldsymbol{\lambda}z.\ (\!|G,z^P,x^P,y^P|\!))$
        & $(\boldsymbol{\lambda}z.\ (\!|F,x^P,z^P,y^P|\!)) = (\boldsymbol{\lambda}z.\ (\!|G,x^P,z^P,y^P|\!))$
        & $(\boldsymbol{\lambda}z.\ (\!|F,x^P,y^P,z^P|\!)) = (\boldsymbol{\lambda}z.\ (\!|G,x^P,y^P,z^P|\!))$
  unfolding *identity-Π_3-def identity-Π_1-def basic-identity$_3$-def forall-ν-def* **by** *simp*
lemma *identity$_\circ$-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-identity$_\circ$-def* **by** *simp*

7 The Axioms of Principia Metaphysica

Remark 17. *The axioms of PM can now be derived from the Semantics and the meta-logic.*

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

7.1 Closures

Remark 18. *The special syntax* [[-]] *is introduced for axioms. This allows to formulate special rules resembling the concepts of closures in PM. To simplify the instantiation of axioms later, special attributes are introduced to automatically resolve the special axiom syntax. Necessitation averse axioms are stated with the syntax for actual validity* [-].

definition *axiom* :: o⇒*bool* ([[-]]) **where** *axiom* ≡ λ φ . ∀ v . [φ *in* v]

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

lemma *axiom-instance*[*axiom*]: [[φ]] ⟹ [φ *in* v]
unfolding *axiom-def* **by** *simp*
lemma *closures-universal*[*axiom*]: (⋀x.[[φ x]]) ⟹ [[∀ x. φ x]]
by *axiom-meta-solver*
lemma *closures-actualization*[*axiom*]: [[φ]] ⟹ [[𝓐 φ]]
by *axiom-meta-solver*
lemma *closures-necessitation*[*axiom*]: [[φ]] ⟹ [[□ φ]]
by *axiom-meta-solver*
lemma *necessitation-averse-axiom-instance*[*axiom*]: [φ] ⟹ [φ *in dw*]
by *meta-solver*
lemma *necessitation-averse-closures-universal*[*axiom*]: (⋀x.[φ x]) ⟹ [∀ x. φ x]
by *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*]:
  $[[\varphi \to (\psi \to \varphi)]]$
  by *axiom-meta-solver*
lemma *pl-2*[*axiom*]:
  $[[(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))]]$
  by *axiom-meta-solver*
lemma *pl-3*[*axiom*]:
  $[[(\neg\varphi \to \neg\psi) \to ((\neg\varphi \to \psi) \to \varphi)]]$
  by *axiom-meta-solver*

7.3 Axioms of Identity

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

7.4 Axioms of Quantification

Remark 19. *The axioms of quantification differ slightly from the axioms in Principia Metaphysica. The differences can be justified, though.*

- *Axiom cqt-2 is omitted, as the embedding does not distinguish between terms and variables. Instead it is combined with cqt-1, in which the corresponding condition is omitted, and with cqt-5 in its modified form cqt-5-mod.*
- *Note that the all quantifier for individuals only ranges over the datatype ν, which is always a denoting term and not a definite description in the embedding.*
- *The case of definite descriptions is handled separately in axiom cqt-1-κ: If a formula on datatype κ holds for all denoting terms ($\forall\ \alpha.\ \varphi\ (\alpha^P)$) then the formula holds for an individual $\varphi\ \alpha$, if α denotes, i.e. $\exists\ \beta\ .\ (\beta^P) = \alpha$.*
- *Although axiom cqt-5 can be stated without modification, it is not a suitable formulation for the embedding. Therefore the seemingly stronger version cqt-5-mod is stated as well. On a closer look, though, cqt-5-mod immediately follows from the original cqt-5 together with the omitted cqt-2.*

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** *blast*
hence $[\varphi\ \alpha$ *in* $v]$
using *l-identity*[**where** φ=φ, *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*)+
moreover hence
$\forall\ w\ .\ ([(\psi\ (\iota x\ .\ \varphi\ x))$ *in* $w] \longrightarrow ($*that* $\varphi) = ($*denotation* $($*that* $\varphi))^P)$
apply *transfer* **by** (*metis* (*mono-tags*, *lifting*) *eq-snd-iff fst-conv option.simps*(*3*))
ultimately show *?thesis*
apply *cut-tac* **unfolding** *identity-κ-def*
apply *axiom-meta-solver* **by** *metis*
qed

lemma *cqt-5-mod*[*axiom*]:
assumes *SimpleExOrEnc* ψ
shows $[[\psi\ x \to (\exists\ \ \alpha\ .\ (\alpha^P) = x)]]$
proof −
have $\forall\ w\ .\ ([(\psi\ x)$ *in* $w] \longrightarrow (\exists\ o_1\ .$ *Some* $o_1 = d_\kappa\ x))$
using *assms* **apply** *induct* **by** (*meta-solver*;*metis*)+
moreover hence $\forall\ w\ .\ ([(\psi\ x)$ *in* $w] \longrightarrow (x) = ($*denotation* $(x))^P)$
apply *transfer* **by** (*metis* (*mono-tags*, *lifting*) *eq-snd-iff fst-conv option.simps*(*3*))
ultimately show *?thesis*
apply *cut-tac* **unfolding** *identity-κ-def*
apply *axiom-meta-solver* **by** *metis*
qed

7.5 Axioms of Actuality

Remark 20. *The necessitation averse axiom of actuality is stated to be actually true; for the statement as a proper axiom (for which necessitation would be allowed) nitpick can find a counter-model as desired.*

lemma *logic-actual*[*axiom*]: $[(\boldsymbol{\mathcal{A}}\varphi) \equiv \varphi]$
apply *meta-solver* **by** *auto*
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 νυ-ων-is-ωυ 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*]:
$[[x^P = (\iota x.\ \varphi\ x) \equiv (\forall z.(\mathcal{A}(\varphi\ z) \equiv z = x))]]$
unfolding *axiom-def*
proof (*rule allI*, *rule EquivI*; *rule*)
fix v
assume $[x^P = (\iota x.\ \varphi\ x)\ in\ v]$
moreover hence *1*:
$\exists\, o_1\ o_2.\ Some\ o_1 = d_\kappa\ (x^P) \wedge Some\ o_2 = d_\kappa\ (\iota x.\ \varphi\ x) \wedge o_1 = o_2$
apply *cut-tac* **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 $\forall\ z\ .\ [\mathcal{A}\varphi\ z\ in\ v] = [(z^P) = (x^P)\ in\ v]$
proof
fix z
have $[\mathcal{A}\varphi\ z\ in\ v] \Longrightarrow [(z^P) = (x^P)\ in\ v]$
unfolding *identity-κ-def* **apply** *meta-solver*
unfolding d_κ*-def* **using** *4 5 2* **apply** *transfer*
apply *simp* **by** (*metis* w_0*-def*)
moreover have $[(z^P) = (x^P)\ in\ v] \Longrightarrow [\mathcal{A}\varphi\ z\ in\ v]$
unfolding *identity-κ-def* **apply** *meta-solver*
using *2 4 5* **apply** *transfer* **apply** *simp*
by (*metis* w_0*-def*)
ultimately show $[\mathcal{A}\varphi\ z\ in\ v] = [(z^P) = (x^P)\ in\ v]$
by *auto*
qed
thus $[\forall z.\ \mathcal{A}\varphi\ z \equiv (z) = (x)\ in\ v]$
unfolding *identity-ν-def*
by (*simp add*: *AllI EquivS*)
next
fix v
assume $[\forall z.\ \mathcal{A}\varphi\ z \equiv (z) = (x)\ in\ v]$
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 *cut-tac* **unfolding** *identity-ν-def identity-κ-def* **by** *meta-solver*
hence $\forall\ z\ .\ evalo\ (\varphi\ z)\ dj\ dw = (z = x)$ **apply** *transfer* **by** *simp*
moreover hence $\exists!x\ .\ evalo\ (\varphi\ x)\ dj\ dw$ **by** *metis*
ultimately have $x^P = (\iota x.\ \varphi\ x)$ **unfolding** *TheS* **by** (*simp add*: *νκ-def*)
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 *IsPropositionalInX* φ
shows $[[(\!|\boldsymbol{\lambda}\ x\ .\ \varphi\ (x^P),\ x^P|\!) \equiv \varphi\ (x^P)]]$
apply *axiom-meta-solver*
using *D5-1*[*OF assms*]
apply *transfer* **by** *simp*

lemma *lambda-predicates-2-2*[*axiom*]:
assumes *IsPropositionalInXY* 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*] **apply** *transfer* **by** *simp*

lemma *lambda-predicates-2-3*[*axiom*]:
assumes *IsPropositionalInXYZ* φ
shows $[[(\!|(\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \varphi\ (x^P)\ (y^P)\ (z^P))),x^P,y^P,z^P|\!) \equiv \varphi\ (x^P)\ (y^P)\ (z^P)]]$
proof −
have $\Box[(\!|(\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \varphi\ (x^P)\ (y^P)\ (z^P))),x^P,y^P,z^P|\!) \rightarrow \varphi\ (x^P)\ (y^P)\ (z^P)]$
apply *meta-solver* **using** *D5-3*[*OF assms*] **by** *auto*
moreover have
$\Box[\varphi\ (x^P)\ (y^P)\ (z^P) \rightarrow (\!|(\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \varphi\ (x^P)\ (y^P)\ (z^P))),x^P,y^P,z^P|\!)]$
apply *axiom-meta-solver*
using *D5-3*[*OF assms*] **unfolding** d_3*-def ex3-def*
apply *transfer* **apply** *simp* **by** *fastforce*
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*]:
$[[(\boldsymbol{\lambda}\ x\ .\ (\!|F,\ x^P|\!)) = F]]$
unfolding *identity-defs*
apply *axiom-meta-solver*
by (*simp add*: *meta-defs meta-aux*)

lemma *lambda-predicates-3-2*[*axiom*]:
$[[(\boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ (\!|F,\ x^P,\ y^P|\!))) = F]]$
unfolding *identity-defs*
apply *axiom-meta-solver*
by (*simp add*: *meta-defs meta-aux*)

lemma *lambda-predicates-3-3*[*axiom*]:
[[($\boldsymbol{\lambda}^3$ (λ *x y z* . ⦇*F*, x^P, y^P, z^P⦈)) = *F*]]
unfolding *identity-defs*
apply *axiom-meta-solver*
by (*simp add*: *meta-defs meta-aux*)

lemma *lambda-predicates-4-0*[*axiom*]:
assumes $\bigwedge x$.[($\boldsymbol{\mathcal{A}}$(φ *x* $\equiv$ ψ *x*)) *in v*]
shows [($\boldsymbol{\lambda}^0$ (χ (ιx. φ *x*)) = $\boldsymbol{\lambda}^0$ (χ (ιx. ψ *x*))) *in v*]
unfolding *identity-defs* **using** *assms* **apply** *cut-tac*
apply *meta-solver* **by** (*auto simp*: *meta-defs*)

lemma *lambda-predicates-4-1*[*axiom*]:
assumes $\bigwedge x$.[($\boldsymbol{\mathcal{A}}$(φ *x* $\equiv$ ψ *x*)) *in v*]
shows [(($\boldsymbol{\lambda}$ *x* . χ (ιx. φ *x*) *x*) = ($\boldsymbol{\lambda}$ *x* . χ (ιx. ψ *x*) *x*)) *in v*]
unfolding *identity-defs* **using** *assms* **apply** *cut-tac*
apply *meta-solver* **by** (*auto simp*: *meta-defs*)

lemma *lambda-predicates-4-2*[*axiom*]:
assumes $\bigwedge x$.[($\boldsymbol{\mathcal{A}}$(φ *x* $\equiv$ ψ *x*)) *in v*]
shows [(($\boldsymbol{\lambda}^2$ (λ *x y* . χ (ιx. φ *x*) *x y*)) = ($\boldsymbol{\lambda}^2$ (λ *x y* . χ (ιx. ψ *x*) *x y*))) *in v*]
unfolding *identity-defs* **using** *assms* **apply** *cut-tac*
apply *meta-solver* **by** (*auto simp*: *meta-defs*)

lemma *lambda-predicates-4-3*[*axiom*]:
assumes $\bigwedge x$.[($\boldsymbol{\mathcal{A}}$(φ *x* $\equiv$ ψ *x*)) *in v*]
shows [($\boldsymbol{\lambda}^3$ (λ *x y z* . χ (ιx. φ *x*) *x y z*)) = ($\boldsymbol{\lambda}^3$ (λ *x y z* . χ (ιx. ψ *x*) *x y z*)) *in v*]
unfolding *identity-defs* **using** *assms* **apply** *cut-tac*
apply *meta-solver* **by** (*auto simp*: *meta-defs*)

7.10 Axioms of Encoding

lemma *encoding*[*axiom*]:
[[⦃*x*,*F*⦄ → □⦃*x*,*F*⦄]]
by *axiom-meta-solver*
lemma *nocoder*[*axiom*]:
[[⦇*O*!,*x*⦈ → ¬(∃ *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*]:
[[∃ *x*. ⦇*A*!,x^P⦈ **&** (∀ *F* . (⦃x^P,*F*⦄ ≡ φ *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 [(∀ *F*. ⦃$?x^P$,*F*⦄ ≡ φ *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 $[(\!|A!, ?x^P|\!) \ \&\ (\forall F.\ \{\!|?x^P, F|\!\} \equiv \varphi\ F)\ \mathit{in}\ v]$
by (*simp only*: *ConjS*)
qed
end

8 Definitions

Various definitions needed throughout PLM.

8.1 Property Negations

consts *propnot* :: $'a \Rightarrow 'a$ (-$^{-}$ [*90*] *90*)
overloading $\mathit{propnot}_0 \equiv \mathit{propnot} :: \Pi_0 \Rightarrow \Pi_0$
$\mathit{propnot}_1 \equiv \mathit{propnot} :: \Pi_1 \Rightarrow \Pi_1$
$\mathit{propnot}_2 \equiv \mathit{propnot} :: \Pi_2 \Rightarrow \Pi_2$
$\mathit{propnot}_3 \equiv \mathit{propnot} :: \Pi_3 \Rightarrow \Pi_3$
begin
definition $\mathit{propnot}_0 :: \Pi_0 \Rightarrow \Pi_0$ **where**
$\mathit{propnot}_0 \equiv \lambda\ p\ .\ \boldsymbol{\lambda}^0\ (\neg p)$
definition $\mathit{propnot}_1$ **where**
$\mathit{propnot}_1 \equiv \lambda\ F\ .\ \boldsymbol{\lambda}\ x\ .\ \neg(\!|F,\ x^P|\!)$
definition $\mathit{propnot}_2$ **where**
$\mathit{propnot}_2 \equiv \lambda\ F\ .\ \boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \neg(\!|F,\ x^P,\ y^P|\!))$
definition $\mathit{propnot}_3$ **where**
$\mathit{propnot}_3 \equiv \lambda\ F\ .\ \boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z\ .\ \neg(\!|F,\ x^P,\ y^P,\ z^P|\!))$
end

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

8.2 Noncontingent and Contingent Relations

consts *Necessary* :: $'a \Rightarrow o$
overloading $\mathit{Necessary}_0 \equiv \mathit{Necessary} :: \Pi_0 \Rightarrow o$
$\mathit{Necessary}_1 \equiv \mathit{Necessary} :: \Pi_1 \Rightarrow o$
$\mathit{Necessary}_2 \equiv \mathit{Necessary} :: \Pi_2 \Rightarrow o$
$\mathit{Necessary}_3 \equiv \mathit{Necessary} :: \Pi_3 \Rightarrow o$
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 *Impossible* :: $'a \Rightarrow o$
overloading $Impossible_0 \equiv Impossible :: \Pi_0 \Rightarrow o$
$Impossible_1 \equiv Impossible :: \Pi_1 \Rightarrow o$
$Impossible_2 \equiv Impossible :: \Pi_2 \Rightarrow o$
$Impossible_3 \equiv Impossible :: \Pi_3 \Rightarrow o$
begin
definition $Impossible_0$ **where**
$Impossible_0 \equiv \lambda\ p\ .\ \Box\neg p$
definition $Impossible_1$ **where**
$Impossible_1 \equiv \lambda\ F\ .\ \Box(\forall\ x.\ \neg(\!|F,x^P|\!))$
definition $Impossible_2$ **where**
$Impossible_2 \equiv \lambda\ F\ .\ \Box(\forall\ x\ y.\ \neg(\!|F,x^P,y^P|\!))$
definition $Impossible_3$ **where**
$Impossible_3 \equiv \lambda\ F\ .\ \Box(\forall\ x\ y\ z.\ \neg(\!|F,x^P,y^P,z^P|\!))$
end

named-theorems *Impossible-defs*
declare $Impossible_0$-*def*[*Impossible-defs*] $Impossible_1$-*def*[*Impossible-defs*]
$Impossible_2$-*def*[*Impossible-defs*] $Impossible_3$-*def*[*Impossible-defs*]

definition *NonContingent* **where**
$NonContingent \equiv \lambda\ F\ .\ (Necessary\ F) \lor (Impossible\ F)$
definition *Contingent* **where**
$Contingent \equiv \lambda\ F\ .\ \neg(Necessary\ F \lor Impossible\ F)$

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

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

8.3 Null and Universal Objects

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

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

8.4 Propositional Properties

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

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*: [[φ]] **for** $\varphi \Rightarrow \langle$*fact A*[*axiom-instance*]$\rangle$)
| *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*]

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*]; [ψ *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*] $\Longrightarrow$ [ψ *in v*]$]\!] \Longrightarrow$ [φ *in v*]
using *CP modus-tollens-2 useful-tautologies-1 vdash-properties-10* **by** *blast*
lemma *reductio-aa-2*[*PLM*]:
$[\![$[φ *in v*] $\Longrightarrow$ [$\neg\psi$ *in v*]; [φ *in v*] $\Longrightarrow$ [ψ *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*]$]\!] \Longrightarrow$ [φ *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*]:
$[\![$[φ *in v*]; [$\neg\psi$ *in v*] $\Longrightarrow$ [$\neg\varphi$ *in v*]$]\!] \Longrightarrow$ ([φ *in v*] $\Longrightarrow$ [ψ *in v*])
using *reductio-aa-1 vdash-properties-9* **by** *blast*
lemma *raa-cor-2*[*PLM*]:
$[\![$[$\neg\varphi$ *in v*]; [$\neg\psi$ *in v*] $\Longrightarrow$ [φ *in v*]$]\!] \Longrightarrow$ ([$\neg\varphi$ *in v*] $\Longrightarrow$ [ψ *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 21. *The classical introduction and elimination rules are proven earlier than in PM. The statements proven so far are sufficient for the proofs and using these rules Isabelle can prove the tautologies 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* ≡*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* ≡*I*)
lemmas ≡*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 ¬¬*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 ¬¬*E* = *intro-elim-8*

context
begin
private lemma *NotNotI*[*PLM-intro*]:
$[\varphi\ in\ v] \Longrightarrow [\neg(\neg\varphi)\ in\ v]$
by (*simp add*: ¬¬*I*)
private lemma *NotNotD*[*PLM-dest*]:
$[\neg(\neg\varphi)\ in\ v] \Longrightarrow [\varphi\ in\ v]$
using ¬¬*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 ∨*I*(*2*) ∨*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*$[PLM]$:
$[\varphi \vee \neg\varphi\ in\ v]$
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*$[PLM]$:
$[(\varphi \vee \varphi) \equiv \varphi\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-e*$[PLM]$:
$[(\varphi \vee \psi) \equiv (\psi \vee \varphi)\ in\ v]$
by *PLM-solver*
lemma *oth-class-taut-3-f*$[PLM]$:
$[(\varphi \vee (\psi \vee \chi)) \equiv ((\varphi \vee \psi) \vee \chi)\ in\ v]$
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*$[PLM]$:

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

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

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

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

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

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

Remark 22. *For the following proofs first the definitions for the respective identities have to be expanded. They are defined directly in the embedded logic, though, so the proofs are still independent of the meta-logic.*

lemma *id-eq-prop-prop-1*[*PLM*]:
$[(F::\Pi_1) = F \text{ in } v]$
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-2*[*PLM*]:
$[((F::\Pi_1) = G) \rightarrow (G = F) \text{ in } v]$
by (*meson id-eq-prop-prop-1 CP ded-thm-cor-3 l-identity*[*axiom-instance*])
lemma *id-eq-prop-prop-3*[*PLM*]:
$[(((F::\Pi_1) = G) \ \&\ (G = H)) \rightarrow (F = H) \text{ in } v]$
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)
lemma *id-eq-prop-prop-4-a*[*PLM*]:
$[(F::\Pi_2) = F \text{ in } v]$
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-4-b*[*PLM*]:
$[(F::\Pi_3) = F \text{ in } v]$
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-5-a*[*PLM*]:
$[((F::\Pi_2) = G) \rightarrow (G = F) \text{ in } v]$
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*]:
$[((F::\Pi_3) = G) \rightarrow (G = F) \text{ in } v]$
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*]:
$[(((F::\Pi_2) = G) \ \&\ (G = H)) \rightarrow (F = H) \text{ in } v]$
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)
lemma *id-eq-prop-prop-6-b*[*PLM*]:
$[(((F::\Pi_3) = G) \ \&\ (G = H)) \rightarrow (F = H) \text{ in } v]$
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)
lemma *id-eq-prop-prop-7*[*PLM*]:
$[(p::\Pi_0) = p \text{ in } v]$
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-7-b*[*PLM*]:
$[(p::\text{o}) = p \text{ in } v]$
unfolding *identity-defs* **by** *PLM-solver*
lemma *id-eq-prop-prop-8*[*PLM*]:
$[((p::\Pi_0) = q) \rightarrow (q = p) \text{ in } v]$
by (*meson id-eq-prop-prop-7 CP ded-thm-cor-3 l-identity*[*axiom-instance*])
lemma *id-eq-prop-prop-8-b*[*PLM*]:
$[((p::\text{o}) = q) \rightarrow (q = p) \text{ in } v]$
by (*meson id-eq-prop-prop-7-b CP ded-thm-cor-3 l-identity*[*axiom-instance*])
lemma *id-eq-prop-prop-9*[*PLM*]:
$[(((p::\Pi_0) = q) \ \&\ (q = r)) \rightarrow (p = r) \text{ in } v]$
by (*metis l-identity*[*axiom-instance*] *ded-thm-cor-4 CP* **&***E*)
lemma *id-eq-prop-prop-9-b*[*PLM*]:
$[(((p::\text{o}) = q) \ \&\ (q = r)) \rightarrow (p = r) \text{ in } v]$
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 (*rule IsPropositional-intros*)
moreover have $[\exists\ \alpha\ .\ (\alpha^P) = x \ in\ v]$
apply (*rule cqt-5-mod*[**where** ψ=$\lambda\ x\ .\ x =_E y$, *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** ψ=$\lambda\ y\ .\ x =_E y$,*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 (*rule IsPropositional-intros*)
moreover have $[\exists\ \alpha\ .\ (\alpha^P) = x \ in\ v]$
apply (*rule cqt-5-mod*[**where** ψ=$\lambda\ x\ .\ (\![O!,x]\!)$,*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 *cut-tac* **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 (*rule IsPropositional-intros*)+
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 *cut-tac* **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 (*rule IsPropositional-intros*)+
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 *cut-tac* **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 23. *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 $x :: \nu$ **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*]:
[$\tau = \tau' \rightarrow (\exists\ \beta\ .\ (\beta^P) = \tau)$ *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 *cut-tac*
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 *cut-tac*
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 *cut-tac*
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 *cut-tac*
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*]:
[(($\boldsymbol{\iota}x.\ \varphi\ x$) = ($\boldsymbol{\iota}x.\ \psi\ x$)) ≡ □(($\boldsymbol{\iota}x.\ \varphi\ x$) = ($\boldsymbol{\iota}x.\ \psi\ x$)) *in v*]
proof (*cases* [(∃ α. (α^P) = ($\boldsymbol{\iota}x$. φ x)) *in v*] ∧ [(∃ β. (β^P) = ($\boldsymbol{\iota}x$. ψ x)) *in v*])
assume [(∃ α. (α^P) = ($\boldsymbol{\iota}x$. φ x)) *in v*] ∧ [(∃ β. (β^P) = ($\boldsymbol{\iota}x$. ψ x)) *in v*]
then obtain α **and** β **where**
[(α^P) = ($\boldsymbol{\iota}x$. φ x) *in v*] ∧ [(β^P) = ($\boldsymbol{\iota}x$. ψ x) *in v*]
apply *cut-tac* **unfolding** *conn-defs* **by** *PLM-solver*
moreover {
moreover have [(α) = (β) ≡ □((α) = (β)) *in v*] **by** *PLM-solver*
ultimately have [(($\boldsymbol{\iota}x.\ \varphi\ x$) = ($\beta^P$) ≡ □(($\boldsymbol{\iota}x.\ \varphi\ x$) = ($\beta^P$))) *in v*]
using *l-identity*[**where** φ=λ α . (α) = (β^P) ≡ □((α) = (β^P)), *axiom-instance*]
modus-ponens **unfolding** *identity-ν-def* **by** *metis*
}
ultimately show *?thesis*
using *l-identity*[**where** φ=λ α . ($\boldsymbol{\iota}x$. φ x) = (α)
≡ □(($\boldsymbol{\iota}x$. φ x) = (α)), *axiom-instance*]
modus-ponens **by** *metis*
next
assume ¬([(∃ α. (α^P) = ($\boldsymbol{\iota}x$. φ x)) *in v*] ∧ [(∃ β. (β^P) = ($\boldsymbol{\iota}x$. ψ x)) *in v*])
hence ¬[⦇A!,($\boldsymbol{\iota}x$. φ x)⦈ *in v*] ∧ ¬[($\boldsymbol{\iota}x$. φ x) $=_E$ ($\boldsymbol{\iota}x$. ψ x) *in v*]
∨ ¬[⦇A!,($\boldsymbol{\iota}x$. ψ x)⦈ *in v*] ∧ ¬[($\boldsymbol{\iota}x$. φ x) $=_E$ ($\boldsymbol{\iota}x$. ψ x) *in v*]
unfolding *identity$_E$-infix-def*
using *cqt-5*[*axiom-instance*] *PLM.contraposition-1 SimpleExOrEnc.intros vdash-properties-10* **by** *meson*
hence ¬[($\boldsymbol{\iota}x$. φ x) = ($\boldsymbol{\iota}x$. ψ x) *in v*]
apply *cut-tac* **unfolding** *identity-defs* **by** *PLM-solver*
thus *?thesis* **apply** *cut-tac* **apply** *PLM-solver*
using *qml-2*[*axiom-instance*, *deduction*] **by** *auto*
qed

9.8 Quantification

— TODO: think about the distinction in PM here
lemma *rule-ui*[*PLM*,*PLM-elim*,*PLM-dest*]:
[∀ α . φ α *in v*] ⟹ [φ β *in v*]
by (*meson cqt-1*[*axiom-instance*, *deduction*])
lemmas ∀ *E* = *rule-ui*

lemma *rule-ui-2*[*PLM*,*PLM-elim*,*PLM-dest*]:
⟦[∀ α . φ (α^P) *in v*]; [∃ α . $(\alpha)^P$ = β *in v*]⟧ ⟹ [φ β *in v*]
using *cqt-1-κ*[*axiom-instance*, *deduction*, *deduction*] **by** *blast*

lemma *cqt-orig-1*[*PLM*]:
[(∀ α. φ α) → φ β *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*]:
$[((\forall\,\alpha.\ \varphi\ \alpha) \vee (\forall\,\alpha.\ \psi\ \alpha)) \rightarrow (\forall\,\alpha.\ (\varphi\ \alpha \vee \psi\ \alpha))\ \ in\ v]$
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*]:
$[(\forall\,\alpha.\ \varphi\ \alpha) \equiv (\forall\,\beta.\ \varphi\ \beta)\ \ in\ v]$
by *PLM-solver*

lemma *existential*[*PLM*,*PLM-intro*]:
$[\varphi\ \alpha\ in\ v] \Longrightarrow [\exists\ \ \alpha.\ \varphi\ \alpha\ in\ v]$
unfolding *exists-def* **by** *PLM-solver*
lemmas $\exists\, I$ = *existential*
lemma *instantiation-*[*PLM*,*PLM-elim*,*PLM-dest*]:
$[\![[\exists\,\alpha\ .\ \varphi\ \alpha\ \ in\ v];\ (\bigwedge\alpha.[\varphi\ \alpha\ in\ v] \Longrightarrow [\psi\ in\ v])]\!] \Longrightarrow [\psi\ \ in\ v]$
unfolding *exists-def* **by** *PLM-solver*

lemma *Instantiate*:
assumes $[\exists\ \ x\ .\ \varphi\ x\ \ \ in\ v]$
obtains x **where** $[\varphi\ x\ in\ v]$

apply (*insert assms*) **unfolding** *exists-def* **by** *PLM-solver*
lemmas $\exists E$ = *Instantiate*

lemma *cqt-further-1*[*PLM*]:
$[(\forall \alpha.\ \varphi\ \alpha) \rightarrow (\exists \alpha.\ \varphi\ \alpha)\ \mathit{in}\ v]$
by *PLM-solver*
lemma *cqt-further-2*[*PLM*]:
$[(\neg(\forall \alpha.\ \varphi\ \alpha)) \equiv (\exists \alpha.\ \neg\varphi\ \alpha)\ \mathit{in}\ v]$
unfolding *exists-def* **by** *PLM-solver*
lemma *cqt-further-3*[*PLM*]:
$[(\forall \alpha.\ \varphi\ \alpha) \equiv \neg(\exists \alpha.\ \neg\varphi\ \alpha)\ \mathit{in}\ v]$
unfolding *exists-def* **by** *PLM-solver*
lemma *cqt-further-4*[*PLM*]:
$[(\neg(\exists \alpha.\ \varphi\ \alpha)) \equiv (\forall \alpha.\ \neg\varphi\ \alpha)\ \mathit{in}\ v]$
unfolding *exists-def* **by** *PLM-solver*
lemma *cqt-further-5*[*PLM*]:
$[(\exists \alpha.\ \varphi\ \alpha\ \&\ \psi\ \alpha) \rightarrow ((\exists \alpha.\ \varphi\ \alpha)\ \&\ (\exists \alpha.\ \psi\ \alpha))\ \mathit{in}\ v]$
unfolding *exists-def* **by** *PLM-solver*
lemma *cqt-further-6*[*PLM*]:
$[(\exists \alpha.\ \varphi\ \alpha \vee \psi\ \alpha) \equiv ((\exists \alpha.\ \varphi\ \alpha) \vee (\exists \alpha.\ \psi\ \alpha))\ \mathit{in}\ v]$
unfolding *exists-def* **by** *PLM-solver*
lemma *cqt-further-10*[*PLM*]:
$[(\varphi\ (\alpha{::}'a{::}\mathit{id\text{-}eq})\ \&\ (\forall\ \beta\ .\ \varphi\ \beta \rightarrow \beta = \alpha)) \equiv (\forall\ \beta\ .\ \varphi\ \beta \equiv \beta = \alpha)$
$\mathit{in}\ v]$
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*]:
$[((\forall \alpha.\ \varphi\ \alpha)\ \&\ (\forall \alpha.\ \psi\ \alpha)) \rightarrow (\forall \alpha.\ \varphi\ \alpha \equiv \psi\ \alpha)\ \mathit{in}\ v]$
by *PLM-solver*
lemma *cqt-further-12*[*PLM*]:
$[((\neg(\exists \alpha.\ \varphi\ \alpha))\ \&\ (\neg(\exists \alpha.\ \psi\ \alpha))) \rightarrow (\forall \alpha.\ \varphi\ \alpha \equiv \psi\ \alpha)\ \mathit{in}\ v]$
unfolding *exists-def* **by** *PLM-solver*
lemma *cqt-further-13*[*PLM*]:
$[((\exists \alpha.\ \varphi\ \alpha)\ \&\ (\neg(\exists \alpha.\ \psi\ \alpha))) \rightarrow (\neg(\forall \alpha.\ \varphi\ \alpha \equiv \psi\ \alpha))\ \mathit{in}\ v]$
unfolding *exists-def* **by** *PLM-solver*
lemma *cqt-further-14*[*PLM*]:
$[(\exists \alpha.\ \exists \beta.\ \varphi\ \alpha\ \beta) \equiv (\exists \beta.\ \exists \alpha.\ \varphi\ \alpha\ \beta)\ \mathit{in}\ v]$
unfolding *exists-def* **by** *PLM-solver*

lemma *nec-exist-unique*[*PLM*]:
$[(\forall\ x.\ \varphi\ x \rightarrow \Box(\varphi\ x)) \rightarrow ((\exists !x.\ \varphi\ x) \rightarrow (\exists !x.\ \Box(\varphi\ x)))\ \mathit{in}\ v]$
proof (*rule CP*)
assume a: $[\forall x.\ \varphi\ x \rightarrow \Box\varphi\ x\ \mathit{in}\ v]$
show $[(\exists !x.\ \varphi\ x) \rightarrow (\exists !x.\ \Box\varphi\ x)\ \mathit{in}\ v]$
proof (*rule CP*)
assume $[(\exists !x.\ \varphi\ x)\ \mathit{in}\ v]$
hence $[\exists \alpha.\ \varphi\ \alpha\ \&\ (\forall \beta.\ \varphi\ \beta \rightarrow \beta = \alpha)\ \mathit{in}\ v]$
by (*simp only*: *exists-unique-def*)
then obtain α **where** *1*:
$[\varphi\ \alpha\ \&\ (\forall \beta.\ \varphi\ \beta \rightarrow \beta = \alpha)\ \mathit{in}\ v]$
by (*rule* $\exists E$)
{

fix β
have $[\Box\varphi\ \beta \rightarrow \beta = \alpha\ in\ v]$
using *1* **&**$E(2)$ *qml-2*[*axiom-instance*]
ded-thm-cor-3 $\forall E$ **by** *fastforce*
}
hence $[\forall\,\beta.\ \Box\varphi\ \beta \rightarrow \beta = \alpha\ in\ v]$ **by** (*rule* $\forall I$)
moreover have $[\Box(\varphi\ \alpha)\ in\ v]$
using *1* **&**$E(1)$ *a vdash-properties-10 cqt-orig-1*[*deduction*]
by *fast*
ultimately have $[\exists\,\alpha.\ \Box(\varphi\ \alpha)\ \&\ (\forall\,\beta.\ \Box\varphi\ \beta \rightarrow \beta = \alpha)\ in\ v]$
using **&**I $\exists I$ **by** *fast*
thus $[(\exists\,!x.\ \Box\varphi\ x)\ in\ v]$
unfolding *exists-unique-def* **by** *assumption*
qed
qed

9.9 Actuality and Descriptions

lemma *nec-imp-act*[*PLM*]: $[\Box\varphi \rightarrow \mathcal{A}\varphi\ in\ v]$
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*]:
$[\mathcal{A}(\mathcal{A}\varphi \rightarrow \varphi)\ in\ v]$
using *equiv-def logic-actual-nec-2*[*axiom-instance*]
logic-actual-nec-4[*axiom-instance*] **&**$E(2)$ $\equiv E(2)$
by *metis*
lemma *act-conj-act-2*[*PLM*]:
$[\mathcal{A}(\varphi \rightarrow \mathcal{A}\varphi)\ in\ v]$
using *logic-actual-nec-2*[*axiom-instance*] *qml-act-1*[*axiom-instance*]
ded-thm-cor-3 $\equiv E(2)$ *nec-imp-act*
by *blast*
lemma *act-conj-act-3*[*PLM*]:
$[(\mathcal{A}\varphi\ \&\ \mathcal{A}\psi) \rightarrow \mathcal{A}(\varphi\ \&\ \psi)\ in\ v]$
unfolding *conn-defs*
by (*metis logic-actual-nec-2*[*axiom-instance*]
logic-actual-nec-1[*axiom-instance*]
$\equiv E(2)$ *CP* $\equiv E(4)$ *reductio-aa-2*
vdash-properties-10)
lemma *act-conj-act-4*[*PLM*]:
$[\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)\ in\ v]$
unfolding *equiv-def*
by (*PLM-solver PLM-intro*: *act-conj-act-3*[**where** $\varphi = \mathcal{A}\varphi \rightarrow \varphi$
and $\psi = \varphi \rightarrow \mathcal{A}\varphi$, *deduction*])
lemma *closure-act-1a*[*PLM*]:
$[\mathcal{A}\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)\ in\ v]$
using *logic-actual-nec-4*[*axiom-instance*]
act-conj-act-4 $\equiv E(1)$
by *blast*
lemma *closure-act-1b*[*PLM*]:
$[\mathcal{A}\mathcal{A}\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)\ in\ v]$
using *logic-actual-nec-4*[*axiom-instance*]

act-conj-act-4 $\equiv E(1)$
by *blast*
lemma *closure-act-1c*[*PLM*]:
$[\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{A}}\varphi \equiv \varphi)\ in\ v]$
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 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*]:
$[\boldsymbol{\mathcal{A}}\varphi \to \boldsymbol{\mathcal{A}}\psi\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}(\varphi \to \psi)\ in\ v]$
using *logic-actual-nec-2*[*axiom-instance*, *equiv-rl*] **.**
private lemma *ActualImplE*[*PLM-dest*, *PLM-elim*]:
$[\boldsymbol{\mathcal{A}}(\varphi \to \psi)\ in\ v] \Longrightarrow [\boldsymbol{\mathcal{A}}\varphi \to \boldsymbol{\mathcal{A}}\psi\ in\ v]$
using *logic-actual-nec-2*[*axiom-instance*, *equiv-lr*] **.**
private lemma *NotActualImplD*[*PLM-dest*]:
$\neg[\boldsymbol{\mathcal{A}}(\varphi \to \psi)\ in\ v] \Longrightarrow \neg[\boldsymbol{\mathcal{A}}\varphi \to \boldsymbol{\mathcal{A}}\psi\ in\ v]$
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*]:
[$\boldsymbol{\mathcal{A}}(\varphi \mathbin{\&} \psi) \equiv (\boldsymbol{\mathcal{A}}\varphi \mathbin{\&} \boldsymbol{\mathcal{A}}\psi)$ *in v*]
by *PLM-solver*
lemma *Act-Basic-3*[*PLM*]:
[$\boldsymbol{\mathcal{A}}(\varphi \equiv \psi) \equiv ((\boldsymbol{\mathcal{A}}(\varphi \rightarrow \psi)) \mathbin{\&} (\boldsymbol{\mathcal{A}}(\psi \rightarrow \varphi)))$ *in v*]
by *PLM-solver*
lemma *Act-Basic-4*[*PLM*]:
[$(\boldsymbol{\mathcal{A}}(\varphi \rightarrow \psi) \mathbin{\&} \boldsymbol{\mathcal{A}}(\psi \rightarrow \varphi)) \equiv (\boldsymbol{\mathcal{A}}\varphi \equiv \boldsymbol{\mathcal{A}}\psi)$ *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*]:
[$\boldsymbol{\mathcal{A}}(\varphi \lor \psi) \equiv \boldsymbol{\mathcal{A}}\varphi \lor \boldsymbol{\mathcal{A}}\psi$ *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 $[\neg\boldsymbol{\mathcal{A}}(\forall\ \alpha\ .\ \neg\varphi\ \alpha) \equiv \neg(\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha)\ \mathit{in}\ v]$
using *oth-class-taut-5-d*[*equiv-lr*] **by** *blast*
moreover have $[\boldsymbol{\mathcal{A}}\neg(\forall\ \alpha\ .\ \neg\varphi\ \alpha) \equiv \neg\boldsymbol{\mathcal{A}}(\forall\ \alpha\ .\ \neg\varphi\ \alpha)\ \mathit{in}\ v]$
using *logic-actual-nec-1*[*axiom-instance*] **by** *blast*
ultimately have $[\boldsymbol{\mathcal{A}}\neg(\forall\ \alpha\ .\ \neg\varphi\ \alpha) \equiv \neg(\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha)\ \mathit{in}\ v]$
using $\equiv E(5)$ **by** *auto*
moreover {
have $[\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha \equiv \neg\boldsymbol{\mathcal{A}}\varphi\ \alpha\ \mathit{in}\ v]$
using *logic-actual-nec-1*[*axiom-universal*, *axiom-instance*] **by** *blast*
hence $[(\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha) \equiv (\forall\ \alpha\ .\ \neg\boldsymbol{\mathcal{A}}\varphi\ \alpha)\ \mathit{in}\ v]$
using *cqt-basic-3*[*deduction*] **by** *fast*
hence $[(\neg(\forall\ \alpha\ .\ \boldsymbol{\mathcal{A}}\neg\varphi\ \alpha)) \equiv \neg(\forall\ \alpha\ .\ \neg\boldsymbol{\mathcal{A}}\varphi\ \alpha)\ \mathit{in}\ v]$
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*]:
$[(\forall\ z\ .\ \boldsymbol{\mathcal{A}}\varphi\ z \equiv z = x) \equiv (\forall\ z\ .\ \varphi\ z \equiv z = x)\ \mathit{in}\ dw]$
by *PLM-solver*

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

lemma *hintikka*[*PLM*]:
$[(x^P = (\boldsymbol{\iota}x.\ \varphi\ x)) \equiv (\varphi\ x\ \&\ (\forall\ z\ .\ \varphi\ z \rightarrow z = x))\ \mathit{in}\ dw]$
proof −
have $[(\forall\ z\ .\ \varphi\ z \equiv z = x) \equiv (\varphi\ x\ \&\ (\forall\ z\ .\ \varphi\ z \rightarrow z = x))\ \mathit{in}\ dw]$
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** $\equiv E(5)$ *fund-cont-desc* **by** *blast*
qed

lemma *russell-axiom-a*[*PLM*]:
$[(\llparenthesis F,\ \boldsymbol{\iota}x.\ \varphi\ x\rrparenthesis) \equiv (\exists\ x\ .\ \varphi\ x\ \&\ (\forall\ z\ .\ \varphi\ z \rightarrow z = x)\ \&\ \llparenthesis F,\ x^P\rrparenthesis)\ \mathit{in}\ dw]$
(**is** [*?lhs* ≡ *?rhs in dw*])
proof −
{
assume *1*: [*?lhs in dw*]
hence $[\exists\,\alpha.\ \alpha^P = (\boldsymbol{\iota}x.\ \varphi\ x)\ \mathit{in}\ dw]$
using *cqt-5*[*axiom-instance*, *deduction*]
SimpleExOrEnc.intros
by *blast*
then obtain α **where** *2*:
$[\alpha^P = (\boldsymbol{\iota}x.\ \varphi\ x)\ \mathit{in}\ dw]$
using $\exists\,E$ **by** *auto*
hence *3*: $[\varphi\ \alpha\ \&\ (\forall\ z\ .\ \varphi\ z \rightarrow z = \alpha)\ \mathit{in}\ dw]$

using *hintikka*[*equiv-lr*] **by** *simp*
from *2* **have** $[(\boldsymbol{\iota}x.\ \varphi\ x) = (\alpha^P)\ \ in\ dw]$
using *l-identity*[**where** α=α^P **and** β=$\boldsymbol{\iota}x.\ \varphi\ x$ **and** φ=$\lambda\ x\ .\ x = \alpha^P$,
axiom-instance, *deduction*, *deduction*]
id-eq-obj-1[**where** x=α] **by** *auto*
hence $[(\!|F,\ \alpha^P|\!)\ in\ dw]$
using *1* *l-identity*[**where** β=α^P **and** α=$\boldsymbol{\iota}x.\ \varphi\ x$ **and** φ=$\lambda\ x\ .\ (\!|F,x|\!)$,
axiom-instance, *deduction*, *deduction*] **by** *auto*
with *3* **have** $[\varphi\ \alpha$ **&** $(\forall\ \ z\ .\ \varphi\ z \to z = \alpha)$ **&** $(\!|F,\ \alpha^P|\!)\ in\ dw]$ **by** (*rule* **&***I*)
hence [*?rhs in dw*] **using** $\exists\,I$[**where** α=α] **by** *simp*
}
moreover {
assume [*?rhs in dw*]
then obtain α **where** *4*:
$[\varphi\ \alpha$ **&** $(\forall\ \ z\ .\ \varphi\ z \to z = \alpha)$ **&** $(\!|F,\ \alpha^P|\!)\ in\ dw]$
using $\exists\,E$ **by** *auto*
hence $[\alpha^P = (\boldsymbol{\iota}x\ .\ \varphi\ x)\ in\ dw] \wedge [(\!|F,\ \alpha^P|\!)\ in\ dw]$
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*]:
$[\{\!|\boldsymbol{\iota}x.\ \varphi\ x,F|\!\} \equiv (\exists\ \ x\ .\ \varphi\ x$ **&** $(\forall\ \ z\ .\ \varphi\ z \to z = x)$ **&** $\{\!|x^P,\ F|\!\})\ in\ dw]$
(**is** [*?lhs* $\equiv$ *?rhs in dw*])
proof –
{
assume *1*: [*?lhs in dw*]
hence $[\exists\,\alpha.\ \alpha^P = (\boldsymbol{\iota}x.\ \varphi\ x)\ in\ dw]$
using *cqt-5*[*axiom-instance*, *deduction*] *SimpleExOrEnc.intros* **by** *blast*
then obtain α **where** *2*: $[\alpha^P = (\boldsymbol{\iota}x.\ \varphi\ x)\ in\ dw]$ **by** (*rule* $\exists\,E$)
hence *3*: $[(\varphi\ \alpha$ **&** $(\forall\ \ z\ .\ \varphi\ z \to z = \alpha))\ in\ dw]$
using *hintikka*[*equiv-lr*] **by** *simp*
from *2* **have** $[(\boldsymbol{\iota}x.\ \varphi\ x) = \alpha^P\ \ in\ dw]$
using *l-identity*[**where** α=α^P **and** β=$\boldsymbol{\iota}x.\ \varphi\ x$ **and** φ=$\lambda\ x\ .\ x = \alpha^P$,
axiom-instance, *deduction*, *deduction*]
id-eq-obj-1[**where** x=α] **by** *auto*
hence $[\{\!|\alpha^P,\ F|\!\}\ in\ dw]$
using *1* *l-identity*[**where** β=α^P **and** α=$\boldsymbol{\iota}x.\ \varphi\ x$ **and** φ=$\lambda\ x\ .\ \{\!|x,F|\!\}$,
axiom-instance, *deduction*, *deduction*] **by** *auto*
with *3* **have** $[(\varphi\ \alpha$ **&** $(\forall\ \ z\ .\ \varphi\ z \to z = \alpha))$ **&** $\{\!|\alpha^P,\ F|\!\}\ in\ dw]$
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:
        [φ α & (∀ z . φ z → z = α) & {|α^P, F|} in dw]
        using ∃E by auto
      hence [α^P = (ιx . φ x) in dw] ∧ [{|α^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 (ιx. φ x) ≡ (∃ x . φ x & (∀ z . φ z → z = x) & ψ (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] assms 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 [ψ (α^P) in dw]
       using 1 l-identity[where β=α^P and α=ιx. φ x and φ=λ x . ψ
x,
                      axiom-instance, deduction, deduction] by auto
      with 3 have [φ α & (∀ z . φ z → z = α) & ψ (α^P) in dw]
        using &I by auto
          hence [?rhs in dw] using ∃I[where α=α] by (simp add:
identity-defs)
    }
    moreover {
      assume [?rhs in dw]
      then obtain α where 4:
        [φ α & (∀ z . φ z → z = α) & ψ (α^P) in dw]
        using ∃E by auto
      hence [α^P = (ιx . φ x) in dw] ∧ [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 ≡E(6) oth-class-taut-4-a)

lemma equi-desc-descA-2[PLM]:
  [(∃ y . y^P = (ιx. φ x)) → ((ιx . φ x) = (ιx . 𝓐φ x)) in v]
  proof (rule CP)
    assume [∃ y. y^P = (ιx. φ x) in v]
    then obtain y where
      [y^P = (ιx. φ x) in v]
      by (rule ∃ E)
    moreover hence [y^P = (ιx. 𝓐φ x) in v]
      using equi-desc-descA-1[equiv-lr] by auto
    ultimately show [(ιx. φ x) = (ιx. 𝓐φ x) in v]
      using l-identity[axiom-instance, deduction, deduction]
      by fast
  qed

lemma equi-desc-descA-3[PLM]:
  assumes SimpleExOrEnc ψ
  shows [ψ (ιx. φ x) → (∃ y . y^P = (ιx. 𝓐φ x)) in v]
  proof (rule CP)
    assume [ψ (ιx. φ x) in v]
    hence [∃ α. α^P = (ιx. φ x) in v]
      using cqt-5[OF assms, axiom-instance, deduction] by auto
    then obtain α where [α^P = (ιx. φ x) in v] by (rule ∃ E)
    hence [α^P = (ιx . 𝓐φ x) in v]
      using equi-desc-descA-1[equiv-lr] by auto
    thus [∃ y. y^P = (ιx. 𝓐φ x) in v]
      using ∃ I by fast
  qed

lemma equi-desc-descA-4[PLM]:
  assumes SimpleExOrEnc ψ
  shows [ψ (ιx. φ x) → ((ιx. φ x) = (ιx. 𝓐φ x)) in v]
  proof (rule CP)
    assume [ψ (ιx. φ x) in v]
    hence [∃ α. α^P = (ιx. φ x) in v]
      using cqt-5[OF assms, axiom-instance, deduction] by auto
    then obtain α where [α^P = (ιx. φ x) in v] by (rule ∃ E)
    moreover hence [α^P  = (ιx . 𝓐φ x) in v]
      using equi-desc-descA-1[equiv-lr] by auto
    ultimately show [(ιx. φ x)  = (ιx . 𝓐φ x) in v]
      using l-identity[axiom-instance, deduction, deduction] by fast
  qed

lemma nec-hintikka-scheme[PLM]:
  [(x^P = (ιx. φ x)) ≡ (𝓐φ x & (∀ z . 𝓐φ z → z = x)) in v]
  using descriptions[axiom-instance]
  apply (rule ≡E(5))
  apply PLM-solver
   using id-eq-obj-1 apply simp
   using id-eq-obj-2[deduction]
         l-identity[where α=x, axiom-instance, deduction, deduction]
   unfolding identity-ν-def
```

```
   apply blast
  using l-identity[where α=x, axiom-instance, deduction, deduction]
  id-eq-2[where 'a=ν, deduction] unfolding identity-ν-def by meson

lemma equiv-desc-eq[PLM]:
  assumes ⋀x.[𝓐(φ x ≡ ψ x) in v]
  shows [(∀ x . ((x^P = (ιx . φ x)) ≡ (x^P = (ιx . ψ x)))) in v]
  proof(rule ∀I)
    fix x
    {
      assume [x^P = (ιx . φ x) in v]
      hence 1: [𝓐φ x & (∀z. 𝓐φ z → z = x) in v]
        using nec-hintikka-scheme[equiv-lr] by auto
      hence 2: [𝓐φ x in v] ∧ [(∀z. 𝓐φ z → z = x) in v]
        using &E by blast
      {
        fix z
        {
          assume [𝓐ψ z in v]
          hence [𝓐φ z in v]
           using assms[where x=z] apply cut-tac by PLM-solver
          moreover have [𝓐φ 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 [𝓐ψ z → z = x in v] by (rule CP)
      }
      hence [(∀ z . 𝓐ψ z → z = x) in v] by (rule ∀I)
      moreover have [𝓐ψ x in v]
        using 1[conj1] assms[where x=x]
        apply cut-tac 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 cut-tac 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 cut-tac 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 $[\exists\,\alpha.\ (\alpha^P) = (\iota x.\ \varphi\ x)\ \mathit{in}\ v]$
  using *cqt-5*[*axiom-instance*, *deduction*] *assms* **by** *blast*
then obtain α **where** *2*: $[(\alpha^P) = (\iota x.\ \varphi\ x)\ \mathit{in}\ v]$ **by** (*rule* $\exists E$)
hence $[(\forall\ z\ .\ (\mathcal{A}(\varphi\ z) \equiv (z = \alpha)))\ \mathit{in}\ v]$
  using *descriptions*[*axiom-instance*, *equiv-lr*] **by** *auto*
hence *3*: $[(\mathcal{A}\varphi\ \alpha)\ \&\ (\forall\ z\ .\ (\mathcal{A}(\varphi\ z) \rightarrow (z = \alpha)))\ \mathit{in}\ v]$
  using *cqt-1*[**where** $\alpha{=}\alpha$ **and** $\varphi{=}\lambda\ z\ .\ (\mathcal{A}(\varphi\ z) \equiv (z = \alpha))$,
    axiom-instance, *deduction*, *equiv-rl*]
  using *id-eq-obj-1*[**where** $x{=}\alpha$] **unfolding** *identity-ν-def*
  using *hintikka*[*equiv-lr*] *cqt-basic-2*[*equiv-lr*,*conj1*]
  $\&I$ **by** *fast*
from *2* **have** $[(\iota x.\ \varphi\ x) = (\alpha^P)\ \ \mathit{in}\ v]$
  using *l-identity*[**where** $\beta{=}(\iota x.\ \varphi\ x)$ **and** $\varphi{=}\lambda\ x\ .\ x = (\alpha^P)$,
    axiom-instance, *deduction*, *deduction*]
    id-eq-obj-1[**where** $x{=}\alpha$] **by** *auto*
hence $[\psi\ (\alpha^P)\ \mathit{in}\ v]$
  using *1* *l-identity*[**where** $\alpha{=}(\iota x.\ \varphi\ x)$ **and** $\varphi{=}\lambda\ x\ .\ \psi\ x$,
    axiom-instance, *deduction*,
    deduction] **by** *auto*
with *3* **have** $[(\mathcal{A}\varphi\ \alpha\ \&\ (\forall\ z\ .\ \mathcal{A}(\varphi\ z) \rightarrow (z = \alpha)))\ \&\ \psi\ (\alpha^P)\ \mathit{in}$
$v]$
  using $\&I$ **by** *simp*
hence [*?rhs in v*]
  using $\exists I$[**where** $\alpha{=}\alpha$]
  by (*simp add*: *identity-defs*)
}
moreover {
assume [*?rhs in v*]
then obtain α **where** *4*:
  $[(\mathcal{A}\varphi\ \alpha\ \&\ (\forall\ z\ .\ \mathcal{A}(\varphi\ z) \rightarrow z = \alpha))\ \&\ \psi\ (\alpha^P)\ \mathit{in}\ v]$
  using $\exists E$ **by** *auto*
hence $[(\forall\ z\ .\ (\mathcal{A}(\varphi\ z) \equiv (z = \alpha)))\ \mathit{in}\ v]$
  using *UniqueAux* $\&E(1)$ **by** *auto*
hence $[(\alpha^P) = (\iota x\ .\ \varphi\ x)\ \mathit{in}\ v] \wedge [\psi\ (\alpha^P)\ \mathit{in}\ v]$
  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*]:
$[(\exists\ y\ .\ (y^P) = (\iota x.\ \varphi\ x)) \equiv (\exists!\ x\ .\ \mathcal{A}(\varphi\ x))\ \mathit{in}\ v]$ (**is** [*?lhs* $\equiv$ *?rhs*
in v])
proof –
{
assume [*?lhs in v*]
then obtain α **where**
  $[((\alpha^P) = (\iota x.\ \varphi\ x))\ \mathit{in}\ v]$
  by (*rule* $\exists E$)
hence $[(\!|A!,(\iota x.\ \varphi\ x)|\!)\ \mathit{in}\ v] \vee [(\alpha^P) =_E (\iota x.\ \varphi\ x)\ \mathit{in}\ v]$

apply (*cut-tac*) **unfolding** *identity-defs* **by** *PLM-solver*
then obtain x **where**
$[((\boldsymbol{\mathcal{A}}\varphi\ x\ \&\ (\forall\ z\ .\ \boldsymbol{\mathcal{A}}(\varphi\ z) \to z = x)))\ in\ v]$
using *nec-russell-axiom*[**where** $\psi{=}\lambda x\ .\ (\!|A!,x|\!)$, *equiv-lr*, *THEN* $\exists E$]
using *nec-russell-axiom*[**where** $\psi{=}\lambda x\ .\ (\alpha^P) =_E x$, *equiv-lr*, *THEN* $\exists E$]
using *SimpleExOrEnc.intros* **unfolding** *identity*$_E$*-infix-def*
by (*meson* **&***E*)
hence [*?rhs in v*] **unfolding** *exists-unique-def* **by** (*rule* $\exists I$)
}
moreover {
assume [*?rhs in v*]
then obtain x **where**
$[((\boldsymbol{\mathcal{A}}\varphi\ x\ \&\ (\forall\ z\ .\ \boldsymbol{\mathcal{A}}(\varphi\ z) \to z = x)))\ in\ v]$
unfolding *exists-unique-def* **by** (*rule* $\exists E$)
hence $[\forall z.\ \boldsymbol{\mathcal{A}}\varphi\ z \equiv z = x\ in\ v]$
using *UniqueAux* **by** *auto*
hence $[(x^P) = (\boldsymbol{\iota}x.\ \varphi\ x)\ in\ v]$
using *descriptions*[*axiom-instance*, *equiv-rl*] **by** *auto*
hence [*?lhs in v*] **by** (*rule* $\exists I$)
}
ultimately show *?thesis*
using $\equiv I$ *CP* **by** *auto*
qed

lemma *actual-desc-2*[*PLM*]:
$[(x^P) = (\boldsymbol{\iota}x.\ \varphi) \to \boldsymbol{\mathcal{A}}\varphi\ in\ v]$
using *nec-hintikka-scheme*[*equiv-lr*, *conj1*]
by (*rule CP*)

lemma *actual-desc-3*[*PLM*]:
$[(z^P) = (\boldsymbol{\iota}x.\ \varphi\ x) \to \boldsymbol{\mathcal{A}}(\varphi\ z)\ in\ v]$
using *nec-hintikka-scheme*[*equiv-lr*, *conj1*]
by (*rule CP*)

lemma *actual-desc-4*[*PLM*]:
$[(\exists\ y\ .\ ((y^P) = (\boldsymbol{\iota}x.\ \varphi\ (x^P)))) \to \boldsymbol{\mathcal{A}}(\varphi\ (\boldsymbol{\iota}x.\ \varphi\ (x^P)))\ in\ v]$
proof (*rule CP*)
assume $[(\exists\ y\ .\ (y^P) = (\boldsymbol{\iota}x\ .\ \varphi\ (x^P)))\ in\ v]$
then obtain y **where** *1*:
$[y^P = (\boldsymbol{\iota}x.\ \varphi\ (x^P))\ in\ v]$
by (*rule* $\exists E$)
hence $[\boldsymbol{\mathcal{A}}(\varphi\ (y^P))\ in\ v]$ **using** *actual-desc-3*[*deduction*] **by** *fast*
thus $[\boldsymbol{\mathcal{A}}(\varphi\ (\boldsymbol{\iota}x.\ \varphi\ (x^P)))\ in\ v]$
using *l-identity*[*axiom-instance*, *deduction*,
deduction] *1* **by** *fast*
qed

lemma *unique-box-desc-1*[*PLM*]:
$[(\exists!x\ .\ \Box(\varphi\ x)) \to (\forall\ y\ .\ (y^P) = (\boldsymbol{\iota}x.\ \varphi\ x) \to \varphi\ y)\ in\ v]$
proof (*rule CP*)
assume $[(\exists!x\ .\ \Box(\varphi\ x))\ in\ v]$
then obtain α **where** *1*:

[$\Box\varphi\ \alpha$ **&** $(\forall\,\beta.\ \Box(\varphi\ \beta) \to \beta = \alpha)$ *in v*]
unfolding *exists-unique-def* **by** (*rule* $\exists$ *E*)
{
fix *y*
{
assume [$(y^P) = (\boldsymbol{\iota}x.\ \varphi\ x)$ *in v*]
hence [$\boldsymbol{\mathcal{A}}\varphi\ \alpha \to \alpha = y$ *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 [$(y^P) = (\boldsymbol{\iota}x.\ \varphi\ x) \to \varphi\ y$ *in v*]
by (*rule CP*)
}
thus [$\forall\ y\ .\ (y^P) = (\boldsymbol{\iota}x.\ \varphi\ x) \to \varphi\ y$ *in v*]
by (*rule* $\forall$ *I*)
qed

lemma *unique-box-desc*[*PLM*]:
[$(\forall\ x\ .\ (\varphi\ x \to \Box(\varphi\ x))) \to ((\exists\,!x\ .\ \varphi\ x)$
$\to (\forall\ y\ .\ (y^P = (\boldsymbol{\iota}x\ .\ \varphi\ x)) \to \varphi\ y))$ *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*]:
$(\bigwedge v.[\varphi \to \psi$ *in v*$]) \Longrightarrow [\Box\varphi \to \Box\psi$ *in v*]
using *RN qml-1*[*axiom-instance*] *vdash-properties-10* **by** *blast*

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

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

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

lemma *KBasic-1*[*PLM*]:
[$\Box\varphi \to \Box(\psi \to \varphi)$ *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]:
    [□(ψ ≡ χ) in v] ⟹ [(¬ψ) ≡ (¬χ) in v]
    using qml-2[axiom-instance] ≡E(1) oth-class-taut-5-d
          vdash-properties-10
    by blast
  lemma rule-sub-lem-1-b[PLM]:
    [□(ψ ≡ χ) in v] ⟹ [(ψ → Θ) ≡ (χ → Θ) in v]
    by (metis equiv-def contraposition-1 CP &E(2) ≡I
             ≡E(1) rule-sub-lem-1-a)
  lemma rule-sub-lem-1-c[PLM]:
    [□(ψ ≡ χ) in v] ⟹ [(Θ → ψ) ≡ (Θ → χ) in v]
    by (metis CP ≡I ≡E(3) ≡E(4) ¬¬I
             ¬¬E rule-sub-lem-1-a)
  lemma rule-sub-lem-1-d[PLM]:
    (⋀x.[□(ψ x ≡ χ x) in v]) ⟹ [(∀α. ψ α) ≡ (∀α. χ α) in v]
    by (metis equiv-def ∀I CP &E ≡I raa-cor-1
             vdash-properties-10 rule-sub-lem-1-a ∀E)
  lemma rule-sub-lem-1-e[PLM]:
    [□(ψ ≡ χ) in v] ⟹ [𝓐ψ ≡ 𝓐χ in v]
    using Act-Basic-5 ≡E(1) nec-imp-act
          vdash-properties-10
    by blast
  lemma rule-sub-lem-1-f[PLM]:
    [□(ψ ≡ χ) in v] ⟹ [□ψ ≡ □χ in v]
    using KBasic-6 ≡I ≡E(1) vdash-properties-9
    by blast

  definition Substable :: (o⇒o) ⇒ bool where
     Substable ≡ λ φ . ∀ ψ χ v . (∀ w . [ψ ≡ χ in w]) ⟶ [φ ψ ≡ φ χ
in v]
  definition Substable1 :: (('a::quantifiable⇒o)⇒o) ⇒ bool where
    Substable1 ≡ λ φ . ∀ ψ χ v . (∀ x w . [ψ x ≡ χ x in w]) ⟶ [φ ψ ≡
φ χ in v]
   definition Substable2 :: (('a::quantifiable⇒'b::quantifiable⇒o)⇒o) ⇒
bool where
    Substable2 ≡ λ φ . ∀ ψ χ v . (∀ x y w . [ψ x y ≡ χ x y in w])
                                  ⟶ [φ ψ ≡ φ χ in v]
  definition SubstableVar :: ((var list⇒o)⇒o) ⇒ bool where
    SubstableVar ≡ λ φ . ∀ ψ χ v . (∀ x w . [ψ x ≡ χ x in w])
                                  ⟶ [φ ψ ≡ φ χ in v]


  lemma rule-sub-nec[PLM]:
    assumes Substable φ
    shows (⋀v.[(ψ ≡ χ) in v]) ⟹ Θ [φ ψ in v] ⟹ Θ [φ χ in v]
    proof −
      assume (⋀v.[(ψ ≡ χ) in v])
      hence [φ ψ in v] = [φ χ in v]
        using assms RN unfolding Substable-def
        using ≡I CP ≡E(1) ≡E(2) by meson
      thus Θ [φ ψ in v] ⟹ Θ [φ χ in v] by auto
```

qed

lemma *rule-sub-nec1*[*PLM*]:
assumes *Substable1* φ
shows $(\bigwedge v\ x\ .[(\psi\ x \equiv \chi\ x)\ in\ v]) \Longrightarrow \Theta\ [\varphi\ \psi\ in\ v] \Longrightarrow \Theta\ [\varphi\ \chi\ in\ v]$
proof –
assume $(\bigwedge v\ x.[(\psi\ x \equiv \chi\ x)\ in\ v])$
hence $[\varphi\ \psi\ in\ v] = [\varphi\ \chi\ in\ v]$
using *assms RN* **unfolding** *Substable1-def*
using $\equiv I$ *CP* $\equiv E(1)$ $\equiv E(2)$ **by** *metis*
thus $\Theta\ [\varphi\ \psi\ in\ v] \Longrightarrow \Theta\ [\varphi\ \chi\ in\ v]$ **by** *auto*
qed

lemma *rule-sub-nec2*[*PLM*]:
assumes *Substable2* φ
shows $(\bigwedge v\ x\ y\ .[\psi\ x\ y \equiv \chi\ x\ y\ in\ v]) \Longrightarrow \Theta\ [\varphi\ \psi\ in\ v] \Longrightarrow \Theta\ [\varphi\ \chi$
$in\ v]$
proof –
assume $(\bigwedge v\ x\ y\ .[\psi\ x\ y \equiv \chi\ x\ y\ in\ v])$
hence $[\varphi\ \psi\ in\ v] = [\varphi\ \chi\ in\ v]$
using *assms RN* **unfolding** *Substable2-def*
using $\equiv I$ *CP* $\equiv E(1)$ $\equiv E(2)$ **by** *metis*
thus $\Theta\ [\varphi\ \psi\ in\ v] \Longrightarrow \Theta\ [\varphi\ \chi\ in\ v]$ **by** *auto*
qed

lemma *rule-sub-necq*[*PLM*]:
assumes *SubstableVar* φ
shows $(\bigwedge v\ x\ .[\psi\ x \equiv \chi\ x\ in\ v]) \Longrightarrow \Theta\ [\varphi\ \psi\ in\ v] \Longrightarrow \Theta\ [\varphi\ \chi\ in\ v]$
proof –
assume $(\bigwedge v\ x.[\psi\ x \equiv \chi\ x\ in\ v])$
hence $[\varphi\ \psi\ in\ v] = [\varphi\ \chi\ in\ v]$
using *assms RN* **unfolding** *SubstableVar-def*
using $\equiv I$ *CP* $\equiv E(1)$ $\equiv E(2)$ **by** *metis*
thus $\Theta\ [\varphi\ \psi\ in\ v] \Longrightarrow \Theta\ [\varphi\ \chi\ in\ v]$ **by** *auto*
qed

definition *SubstableAuxVar* :: $('a \Rightarrow (var\ list \Rightarrow o) \Rightarrow (var\ list \Rightarrow o)) \Rightarrow bool$
where
SubstableAuxVar $\equiv \lambda\ \varphi\ .\ \forall\ \psi\ \chi\ v\ x\ bndvars\ .\ (\forall\ x\ v\ .\ [\psi\ x \equiv \chi\ x\ in$
$v])$
$\longrightarrow ([\varphi\ bndvars\ \psi\ x \equiv \varphi\ bndvars\ \chi\ x\ in\ v])$

named-theorems *Substable-intros*

lemma *SubstableVar-intro*[*Substable-intros*]:
SubstableAuxVar $\psi \Longrightarrow$ *SubstableVar* $(\lambda\ \varphi\ .\ \psi\ (\Theta\ x)\ \varphi\ x)$
unfolding *SubstableVar-def SubstableAuxVar-def* **by** *blast*
lemma *SubstableAux-bndvars-intro*[*Substable-intros*]:
SubstableAuxVar $(\lambda\ bndvars\ \varphi\ x\ .\ \varphi\ (\Theta\ bndvars\ x))$
unfolding *SubstableAuxVar-def* **using** *qml-2*[*axiom-instance*, *deduction*] **by** *blast*
lemma *SubstableAux-const-intro*[*Substable-intros*]:
SubstableAuxVar $(\lambda\ bndvars\ \varphi\ x\ .\ \Theta\ bndvars\ x)$
unfolding *SubstableAuxVar-def* **using** *oth-class-taut-4-a* **by** *blast*

lemma *SubstableAux-not-intro*[*Substable-intros*]:
SubstableAuxVar $\psi \Longrightarrow$ *SubstableAuxVar* (λ *bndvars* φ *x*.
$\neg$(ψ (Θ*1 bndvars x*) φ (Θ*2 bndvars x*)))
unfolding *SubstableAuxVar-def*
using *rule-sub-lem-1-a RN-2* $\equiv$*E*(*1*) *oth-class-taut-5-d* **by** *blast*
lemma *SubstableAux-impl-intro*[*Substable-intros*]:
SubstableAuxVar $\psi \Longrightarrow$ *SubstableAuxVar* $\chi \Longrightarrow$ *SubstableAuxVar* (λ *bndvars* φ *x*.
(ψ (Θ*1 bndvars x*) φ (Θ*2 bndvars x*)) $\rightarrow$ (χ (Θ*3 bndvars x*) φ (Θ*4 bndvars x*)))
unfolding *SubstableAuxVar-def* **by** (*metis* $\equiv$*I CP intro-elim-6-a intro-elim-6-b*)
lemma *SubstableAux-box-intro*[*Substable-intros*]:
SubstableAuxVar $\psi \Longrightarrow$ *SubstableAuxVar* (λ *bndvars* φ *x*.
$\Box$(ψ (Θ*1 bndvars x*) φ (Θ*2 bndvars x*)))
unfolding *SubstableAuxVar-def* **using** *rule-sub-lem-1-f RN* **by** *meson*
lemma *SubstableAux-actual-intro*[*Substable-intros*]:
SubstableAuxVar $\psi \Longrightarrow$ *SubstableAuxVar* (λ *bndvars* φ *x*.
$\boldsymbol{\mathcal{A}}$(ψ (Θ*1 bndvars x*) φ (Θ*2 bndvars x*)))
unfolding *SubstableAuxVar-def* **using** *rule-sub-lem-1-e RN* **by** *meson*
lemma *SubstableAux-all-intro*[*Substable-intros*]:
SubstableAuxVar $\psi \Longrightarrow$ *SubstableAuxVar* (λ *bndvars* φ *x*.
$\forall$ *y* . (ψ (Θ*1 bndvars x y*) φ (Θ*2 bndvars x y*)))
unfolding *SubstableAuxVar-def*
proof (*rule allI*)+
fix Ψ χ :: *var list*$\Rightarrow$o **and** *v x bndvars*
assume *a1*: $\forall\,\Psi\ \chi$ *v x bndvars*. ($\forall$ *x w* . [Ψ *x* $\equiv$ χ *x in w*])
$\longrightarrow$ [ψ *bndvars* Ψ *x* $\equiv$ ψ *bndvars* χ *x in v*]
{
assume *a2*: ($\forall$ *x v*. [Ψ *x* $\equiv$ χ *x in v*])
{
fix *y*
have [ψ (Θ*1 bndvars x y*) Ψ (Θ*2 bndvars x y*)
$\equiv$ ψ (Θ*1 bndvars x y*) χ (Θ*2 bndvars x y*) *in v*]
using *a1 a2* **by** *auto*
}
hence [($\forall$ *y*. ψ (Θ*1 bndvars x y*) Ψ (Θ*2 bndvars x y*))
$\equiv$ ($\forall$ *y*. ψ (Θ*1 bndvars x y*) χ (Θ*2 bndvars x y*)) *in v*]
using *cqt-basic-3*[*deduction*] $\forall$*I* **by** *fast*
}
thus ($\forall$ *x v* . [Ψ *x* $\equiv$ χ *x in v*]) $\longrightarrow$
[($\forall$ *y*. ψ (Θ*1 bndvars x y*) Ψ (Θ*2 bndvars x y*))
$\equiv$ ($\forall$ *y*. ψ (Θ*1 bndvars x y*) χ (Θ*2 bndvars x y*)) *in v*]
by *auto*
qed

lemma *Substable-intro*[*Substable-intros*]:
SubstableVar (λ φ . ψ φ) $\Longrightarrow$ *Substable* (λ φ . ψ (λ *v* . φ))
unfolding *SubstableVar-def Substable-def* **by** *fast*

lemma *Substable1-intro*[*Substable-intros*]:
SubstableVar (λ φ . ψ (λ *y* . φ ((*qvar y*)#*Nil*))) $\Longrightarrow$ *Substable1* ψ
unfolding *SubstableVar-def Substable1-def*
proof (*rule allI*)+

fix Ψ :: *'a::quantifiable*$\Rightarrow$o **and** χ v
assume *1*: $\forall$ Ψ χ v.
($\forall x\ w$. [Ψ $x \equiv \chi$ x *in* w]) $\longrightarrow$ [ψ (λy. Ψ ((*qvar* y)#*Nil*))
$\equiv \psi$ (λy. χ ((*qvar* y)#*Nil*)) *in* v]
{
assume ($\forall x\ w$. [Ψ $x \equiv \chi$ x *in* w])
hence [ψ (λy. Ψ (*varq* (*hd* ((*qvar* y)#*Nil*))))
$\equiv \psi$ (λ y . χ (*varq* (*hd* ((*qvar* y)#*Nil*)))) *in* v]
using *1* **by** *fast*
hence [ψ (λy. Ψ y) $\equiv \psi$ (λ y . χ y) *in* v]
using *varq-qvar-id*[**where** *'a='a*] **by** *fastforce*
}
thus ($\forall x\ w$. [Ψ $x \equiv \chi$ x *in* w]) $\longrightarrow$ [ψ $\Psi \equiv \psi$ χ *in* v]
by *blast*
qed

lemma *Substable2-intro*[*Substable-intros*]:
SubstableVar (λ φ . ψ (λ x y . φ ((*qvar* x)#(*qvar* y)#*Nil*))) $\Longrightarrow$ *Substable2* ψ
unfolding *SubstableVar-def Substable2-def*
proof (*rule allI*)+
fix Ψ :: *'a::quantifiable*$\Rightarrow$*'b::quantifiable*$\Rightarrow$o **and** χ v
let *?L* = λ x y . (*qvar* x)#(*qvar* y)#*Nil*
assume *1*: $\forall$ Ψ χ v. ($\forall x\ w$. [Ψ $x \equiv \chi$ x *in* w])
$\longrightarrow$ [ψ ($\lambda x\ y$. Ψ (*?L* x y)) $\equiv \psi$ ($\lambda x\ y$. χ (*?L* x y)) *in* v]
{
assume $\forall x\ y\ w$. [Ψ x $y \equiv \chi$ x y *in* w]
hence [ψ ($\lambda x\ y$. Ψ (*varq* (*hd* (*?L* x y))) (*varq* (*hd* (*tl* (*?L* x y)))))
$\equiv \psi$ (λx y . χ (*varq* (*hd* (*?L* x y))) (*varq* (*hd* (*tl* (*?L* x y))))) *in* v]
using *1* **by** *fast*
hence [ψ ($\lambda x\ y$. Ψ x y) $\equiv \psi$ (λx y . χ x y) *in* v]
using *varq-qvar-id*[**where** *'a='a*] *varq-qvar-id*[**where** *'a='b*] **by** *fastforce*
}
thus ($\forall x\ y\ w$. [Ψ x $y \equiv \chi$ x y *in* w]) $\longrightarrow$ [ψ $\Psi \equiv \psi$ χ *in* v]
by *blast*
qed

lemma *SubstableAux-conj-intro*[*Substable-intros*]:
SubstableAuxVar ψ $\Longrightarrow$ *SubstableAuxVar* χ $\Longrightarrow$ *SubstableAuxVar* (λ *bndvars* φ x.
(ψ (Θ*1* *bndvars* x) φ (Θ*2* *bndvars* x)) **&** (χ (Θ*3* *bndvars* x) φ (Θ*5* *bndvars* x)))
unfolding *conn-defs* **by** ((*rule Substable-intros*)+; ((*assumption*+)*?*))+
lemma *SubstableAux-disj-intro*[*Substable-intros*]:
SubstableAuxVar ψ $\Longrightarrow$ *SubstableAuxVar* χ $\Longrightarrow$ *SubstableAuxVar* (λ *bndvars* φ x.
(ψ (Θ*1* *bndvars* x) φ (Θ*2* *bndvars* x)) $\vee$ (χ (Θ*3* *bndvars* x) φ (Θ*4* *bndvars* x)))
unfolding *conn-defs* **by** ((*rule Substable-intros*)+; ((*assumption*+)*?*))+
lemma *SubstableAux-equiv-intro*[*Substable-intros*]:
SubstableAuxVar ψ $\Longrightarrow$ *SubstableAuxVar* χ $\Longrightarrow$ *SubstableAuxVar* (λ

```
bndvars φ x.
      (ψ (Θ1 bndvars x) φ (Θ2 bndvars x)) ≡ (χ (Θ3 bndvars x) φ (Θ4
bndvars x)))
  unfolding conn-defs by ((rule Substable-intros)+; ((assumption+)?))+
 lemma SubstableAux-diamond-intro[Substable-intros]:
   SubstableAuxVar ψ ⟹ SubstableAuxVar (λ bndvars φ x.
     ◇(ψ (Θ1 bndvars x) φ (Θ2 bndvars x)))
  unfolding conn-defs by ((rule Substable-intros)+; ((assumption+)?))+
 lemma SubstableAux-exists-intro[Substable-intros]:
   SubstableAuxVar ψ ⟹ SubstableAuxVar (λ bndvars φ x.
     ∃ y . (ψ (Θ1 bndvars x y) φ (Θ2 bndvars x y)))
  unfolding conn-defs by ((rule Substable-intros)+; ((assumption+)?))+

 method PLM-subst-method for ψ::o and χ::o =
   (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],
        ((rule Substable-intros, ((assumption)+)?)+; fail))›)
 method PLM-subst-goal-method for φ::o⇒o and ψ::o =
   (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],
        ((rule Substable-intros, ((assumption)+)?)+; fail))›)
 method PLM-subst1-method for ψ::('a::quantifiable)⇒o and χ::('a)⇒o
=
   (match conclusion in Θ [φ χ in v] for Θ and φ and v ⇒
     ‹(rule rule-sub-nec1[where Θ=Θ and χ=χ and ψ=ψ and φ=φ and
v=v],
        ((rule Substable-intros, ((assumption)+)?)+; fail))›)
  method PLM-subst1-goal-method for φ::('a::quantifiable⇒o)⇒o and
ψ::'a⇒o =
   (match conclusion in Θ [φ χ in v] for Θ and χ and v ⇒
     ‹(rule rule-sub-nec1[where Θ=Θ and χ=χ and ψ=ψ and φ=φ and
v=v],
        ((rule Substable-intros, ((assumption)+)?)+; fail))›)
 method PLM-subst2-method for ψ::'a::quantifiable⇒'a⇒o and χ::'a⇒'a⇒o
=
   (match conclusion in Θ [φ χ in v] for Θ and φ and v ⇒
     ‹(rule rule-sub-nec2[where Θ=Θ and χ=χ and ψ=ψ and φ=φ and
v=v],
        ((rule Substable-intros, ((assumption)+)?)+; fail))›)
 method PLM-subst2-goal-method for φ::('a::quantifiable⇒'a⇒o)⇒o
                              and ψ::'a⇒'a⇒o =
   (match conclusion in Θ [φ χ in v] for Θ and χ and v ⇒
     ‹(rule rule-sub-nec2[where Θ=Θ and χ=χ and ψ=ψ and φ=φ and
v=v],
        ((rule Substable-intros, ((assumption)+)?)+; fail))›)

 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],
          ((rule Substable-intros, ((assumption)+)?)+; fail))› ›)
```

method *PLM-autosubst-with* **uses** *WITH* =
($match$ $WITH$ **in** Y: $\bigwedge v$. [$\psi \equiv \chi$ *in* v] **for** ψ **and** $\chi \Rightarrow$
⟨ *match conclusion in* Θ [φ χ *in* v] *for* Θ *and* φ *and* $v \Rightarrow$
⟨(*rule rule-sub-nec*[*where* Θ=Θ *and* χ=χ *and* ψ=ψ *and* φ=φ *and* v=v],
((*rule Substable-intros*)+; *fail*)), ((*fact WITH*)?)⟩ ⟩)
method *PLM-autosubst1* =
(*match* **premises in** $\bigwedge v$ x :: $'a$::*quantifiable* . [ψ $x \equiv \chi$ x *in* v] **for** ψ **and** $\chi \Rightarrow$
⟨ *match conclusion in* Θ [φ χ *in* v] *for* Θ *and* φ *and* $v \Rightarrow$
⟨(*rule rule-sub-nec1*[*where* Θ=Θ *and* χ=χ *and* ψ=ψ *and* φ=φ *and* v=v],
((*rule Substable-intros*, ((*assumption*)+)?)+; *fail*))⟩ ⟩)
method *PLM-autosubst2* =
(*match* **premises in** $\bigwedge v$ (x :: $'a$::*quantifiable*) (y::$'a$) . [ψ x $y \equiv \chi$ x y *in* v]
for ψ **and** $\chi \Rightarrow$
⟨ *match conclusion in* Θ [φ χ *in* v] *for* Θ *and* φ *and* $v \Rightarrow$
⟨(*rule rule-sub-nec2*[*where* Θ=Θ *and* χ=χ *and* ψ=ψ *and* φ=φ *and* v=v],
((*rule Substable-intros*, ((*assumption*)+)?)+; *fail*))⟩ ⟩)

lemma *rule-sub-remark-1*:
assumes ($\bigwedge v$.[(|$A!$,x|) $\equiv$ ($\neg$($\Diamond$(|$E!$,x|))) *in* v])
and [$\neg$(|$A!$,x|) *in* v]
shows[$\neg\neg\Diamond$(|$E!$,x|) *in* v]
apply (*insert assms*) **apply** *PLM-autosubst* **by** *auto*

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

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

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

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

lemma *rule-sub-remark-6*:
assumes $\bigwedge v$.[$\psi \equiv \chi$ *in* v]

and $[\Box(\varphi \rightarrow \psi)$ *in* $v]$
shows $[\Box(\varphi \rightarrow \chi)$ *in* $v]$
apply (*insert assms*) **apply** *PLM-autosubst* **by** *auto*

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

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

lemma *rule-sub-remark-9*:
assumes $\bigwedge v.[(\!|P,a|\!) \equiv ((\!|P,a|\!)$ **&** $((\!|Q,b|\!) \vee (\neg(\!|Q,b|\!))))$ *in* $v]$
and $[(\!|P,a|\!) = (\!|P,a|\!)$ *in* $v]$
shows $[(\!|P,a|\!) = ((\!|P,a|\!)$ **&** $((\!|Q,b|\!) \vee (\neg(\!|Q,b|\!))))$ *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*]:
  $[\Diamond(\varphi \vee \psi) \equiv (\Diamond\varphi \vee \Diamond\psi)$ *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 *cut-tac* **apply** (*PLM-subst-method* $\Box(\neg\varphi)$ $\neg(\Diamond\varphi)$)
       apply (*rule KBasic2-4*)
      apply (*PLM-subst-method* $\Box(\neg\psi)$ $\neg(\Diamond\psi)$)
       apply (*rule KBasic2-4*)
      unfolding *diamond-def* **by** *assumption*
    hence $[(\neg(\Diamond(\varphi \vee \psi))) \equiv ((\neg(\Diamond\varphi))$ **&** $(\neg(\Diamond\psi)))$ *in* $v]$
     apply *cut-tac* **apply** (*PLM-subst-method* $\neg((\neg\varphi)$ **&** $(\neg\psi))$ $\varphi \vee \psi$)
      using *oth-class-taut-6-b*[*equiv-sym*] **by** *auto*
    hence $[(\neg(\neg(\Diamond(\varphi \vee \psi)))) \equiv (\neg((\neg(\Diamond\varphi))$**&**$(\neg(\Diamond\psi))))$ *in* $v]$
      by (*rule oth-class-taut-5-d*[*equiv-lr*])
    hence $[\Diamond(\varphi \vee \psi) \equiv (\neg((\neg(\Diamond\varphi))$ **&** $(\neg(\Diamond\psi))))$ *in* $v]$
      apply *cut-tac* **apply** (*PLM-subst-method* $\neg(\neg(\Diamond(\varphi \vee \psi)))$ $\Diamond(\varphi \vee$
$\psi)$)
      using *oth-class-taut-4-b*[*equiv-sym*] **by** *assumption+*
    thus *?thesis*
      apply *cut-tac* **apply** (*PLM-subst-method* $\neg((\neg(\Diamond\varphi))$ **&** $(\neg(\Diamond\psi)))$
$(\Diamond\varphi) \vee (\Diamond\psi)$)
      using *oth-class-taut-6-b*[*equiv-sym*] **by** *assumption+*
  qed

lemma *KBasic2-7*[*PLM*]:
  $[(\Box\varphi \vee \Box\psi) \rightarrow \Box(\varphi \vee \psi)$ *in* $v]$
  proof −
    have $\bigwedge v$. $[\varphi \rightarrow (\varphi \vee \psi)$ *in* $v]$
        by (*metis contraposition-1 contraposition-2 useful-tautologies-3*
disj-def)
    hence $[\Box\varphi \rightarrow \Box(\varphi \vee \psi)$ *in* $v]$ **using** *RM-1* **by** *auto*
    moreover **{**
        have $\bigwedge v$. $[\psi \rightarrow (\varphi \vee \psi)$ *in* $v]$
          by (*simp only*: *pl-1*[*axiom-instance*] *disj-def*)
        hence $[\Box\psi \rightarrow \Box(\varphi \vee \psi)$ *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*]:
  $[\Diamond(\varphi$ **&** $\psi) \rightarrow (\Diamond\varphi$ **&** $\Diamond\psi)$ *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*]:
  $[\Diamond(\varphi \rightarrow \psi) \equiv (\Box\varphi \rightarrow \Diamond\psi)$ *in* $v]$
  apply (*PLM-subst-method* $(\neg(\Box\varphi)) \vee (\Diamond\psi)$ $\Box\varphi \rightarrow \Diamond\psi$)

using *oth-class-taut-5-k*[*equiv-sym*] **apply** *assumption*
apply (*PLM-subst-method* $(\neg\varphi) \lor \psi\ \varphi \to \psi$)
using *oth-class-taut-5-k*[*equiv-sym*] **apply** *assumption*
apply (*PLM-subst-method* $\Diamond(\neg\varphi)\ \neg(\Box\varphi)$)
using *KBasic2-2*[*equiv-sym*] **apply** *assumption*
using *KBasic2-6* **.**

lemma *KBasic2-10*[*PLM*]:
$[\Diamond(\Box\varphi) \equiv (\neg(\Box\Diamond(\neg\varphi)))$ *in* $v]$
unfolding *diamond-def* **apply** (*PLM-subst-method* $\varphi\ \neg\neg\varphi$)
using *oth-class-taut-4-b oth-class-taut-4-a* **by** *auto*

lemma *KBasic2-11*[*PLM*]:
$[\Diamond\Diamond\varphi \equiv (\neg(\Box\Box(\neg\varphi)))$ *in* $v]$
unfolding *diamond-def*
apply (*PLM-subst-method* $\Box(\neg\varphi)\ \neg(\neg(\Box(\neg\varphi)))$)
using *oth-class-taut-4-b oth-class-taut-4-a* **by** *auto*

lemma *KBasic2-12*[*PLM*]: $[\Box(\varphi \lor \psi) \to (\Box\varphi \lor \Diamond\psi)$ *in* $v]$
proof −
have $[\Box(\psi \lor \varphi) \to (\Box(\neg\psi) \to \Box\varphi)$ *in* $v]$
using *CP RM-1-b* $\lor E(2)$ **by** *blast*
hence $[\Box(\psi \lor \varphi) \to (\Diamond\psi \lor \Box\varphi)$ *in* $v]$
unfolding *diamond-def disj-def*
by (*meson CP* $\neg\neg E$ *vdash-properties-6*)
thus *?thesis* **apply** *cut-tac*
apply (*PLM-subst-method* $(\Diamond\psi \lor \Box\varphi)\ (\Box\varphi \lor \Diamond\psi)$)
apply (*simp add*: *PLM.oth-class-taut-3-e*)
apply (*PLM-subst-method* $(\psi \lor \varphi)\ (\varphi \lor \psi)$)
apply (*simp add*: *PLM.oth-class-taut-3-e*)
by *assumption*
qed

lemma *TBasic*[*PLM*]:
$[\varphi \to \Diamond\varphi$ *in* $v]$
unfolding *diamond-def*
apply (*subst contraposition-1*)
apply (*PLM-subst-method* $\Box\neg\varphi\ \neg\neg\Box\neg\varphi$)
apply (*simp only*: *PLM.oth-class-taut-4-b*)
using *qml-2*[**where** $\varphi=\neg\varphi$, *axiom-instance*]
by *assumption*
lemmas $T\Diamond$ = *TBasic*

lemma *S5Basic-1*[*PLM*]:
$[\Diamond\Box\varphi \to \Box\varphi$ *in* $v]$
proof (*rule CP*)
assume $[\Diamond\Box\varphi$ *in* $v]$
hence $[\neg\Box\Diamond\neg\varphi$ *in* $v]$
using *KBasic2-10*[*equiv-lr*] **by** *simp*
moreover have $[\Diamond(\neg\varphi) \to \Box\Diamond(\neg\varphi)$ *in* $v]$
by (*simp add*: *qml-3*[*axiom-instance*])
ultimately have $[\neg\Diamond\neg\varphi$ *in* $v]$
by (*simp add*: *PLM.modus-tollens-1*)
thus $[\Box\varphi$ *in* $v]$

unfolding *diamond-def* **apply** *cut-tac*
apply (*PLM-subst-method* $\neg\neg\varphi$ φ)
using *oth-class-taut-4-b*[*equiv-sym*] **apply** *assumption*
unfolding *diamond-def* **using** *oth-class-taut-4-b*[*equiv-rl*]
by *simp*
qed
lemmas $5\Diamond$ = *S5Basic-1*

lemma *S5Basic-2*[*PLM*]:
[$\Box\varphi \equiv \Diamond\Box\varphi$ *in v*]
using $5\Diamond$ $T\Diamond$ $\equiv I$ **by** *blast*

lemma *S5Basic-3*[*PLM*]:
[$\Diamond\varphi \equiv \Box\Diamond\varphi$ *in v*]
using *qml-3*[*axiom-instance*] *qml-2*[*axiom-instance*] $\equiv I$ **by** *blast*

lemma *S5Basic-4*[*PLM*]:
[$\varphi \rightarrow \Box\Diamond\varphi$ *in v*]
using $T\Diamond$[*deduction*, *THEN S5Basic-3*[*equiv-lr*]]
by (*rule CP*)

lemma *S5Basic-5*[*PLM*]:
[$\Diamond\Box\varphi \rightarrow \varphi$ *in v*]
using *S5Basic-2*[*equiv-rl*, *THEN qml-2*[*axiom-instance*, *deduction*]]
by (*rule CP*)
lemmas $B\Diamond$ = *S5Basic-5*

lemma *S5Basic-6*[*PLM*]:
[$\Box\varphi \rightarrow \Box\Box\varphi$ *in v*]
using *S5Basic-4*[*deduction*] *RM-1*[*OF S5Basic-1*, *deduction*] *CP* **by**
auto
lemmas $4\Box$ = *S5Basic-6*

lemma *S5Basic-7*[*PLM*]:
[$\Box\varphi \equiv \Box\Box\varphi$ *in v*]
using $4\Box$ *qml-2*[*axiom-instance*] **by** (*rule* $\equiv I$)

lemma *S5Basic-8*[*PLM*]:
[$\Diamond\Diamond\varphi \rightarrow \Diamond\varphi$ *in v*]
using *S5Basic-6*[**where** φ=$\neg\varphi$, *THEN contraposition-1*[*THEN iffD1*],
deduction]
KBasic2-11[*equiv-lr*] *CP* **unfolding** *diamond-def* **by** *auto*
lemmas $4\Diamond$ = *S5Basic-8*

lemma *S5Basic-9*[*PLM*]:
[$\Diamond\Diamond\varphi \equiv \Diamond\varphi$ *in v*]
using $4\Diamond$ $T\Diamond$ **by** (*rule* $\equiv I$)

lemma *S5Basic-10*[*PLM*]:
[$\Box(\varphi \vee \Box\psi) \equiv (\Box\varphi \vee \Box\psi)$ *in v*]
apply (*rule* $\equiv I$)
apply (*PLM-subst-goal-method* $\lambda\,\chi\,.\,\Box(\varphi \vee \Box\psi) \rightarrow (\Box\varphi \vee \chi)$ $\Diamond\Box\psi$)
using *S5Basic-2*[*equiv-sym*] **apply** *assumption*
using *KBasic2-12* **apply** *assumption*

apply (*PLM-subst-goal-method* $\lambda\ \chi\ .(\Box\varphi \vee \chi) \rightarrow \Box(\varphi \vee \Box\psi)\ \Box\Box\psi$)
using *S5Basic-7*[*equiv-sym*] **apply** *assumption*
using *KBasic2-7* **by** *auto*

lemma *S5Basic-11*[*PLM*]:
[$\Box(\varphi \vee \Diamond\psi) \equiv (\Box\varphi \vee \Diamond\psi)$ *in v*]
apply (*rule* $\equiv$*I*)
apply (*PLM-subst-goal-method* $\lambda\ \chi\ .\ \Box(\varphi \vee \Diamond\psi) \rightarrow (\Box\varphi \vee \chi)\ \Diamond\Diamond\psi$)
using *S5Basic-9* **apply** *assumption*
using *KBasic2-12* **apply** *assumption*
apply (*PLM-subst-goal-method* $\lambda\ \chi\ .(\Box\varphi \vee \chi) \rightarrow \Box(\varphi \vee \Diamond\psi)\ \Box\Diamond\psi$)
using *S5Basic-3*[*equiv-sym*] **apply** *assumption*
using *KBasic2-7* **by** *assumption*

lemma *S5Basic-12*[*PLM*]:
[$\Diamond(\varphi\ \&\ \Diamond\psi) \equiv (\Diamond\varphi\ \&\ \Diamond\psi)$ *in v*]
proof −
have [$\Box((\neg\varphi) \vee \Box(\neg\psi)) \equiv (\Box(\neg\varphi) \vee \Box(\neg\psi))$ *in v*]
using *S5Basic-10* **by** *auto*
hence *1*: [$(\neg\Box((\neg\varphi) \vee \Box(\neg\psi))) \equiv \neg(\Box(\neg\varphi) \vee \Box(\neg\psi))$ *in v*]
using *oth-class-taut-5-d*[*equiv-lr*] **by** *auto*
have *2*: [$(\Diamond(\neg((\neg\varphi) \vee (\neg(\Diamond\psi))))) \equiv (\neg((\neg(\Diamond\varphi)) \vee (\neg(\Diamond\psi))))$ *in*
v]
apply (*PLM-subst-method* $\Box\neg\psi\ \neg\Diamond\psi$)
using *KBasic2-4* **apply** *assumption*
apply (*PLM-subst-method* $\Box\neg\varphi\ \neg\Diamond\varphi$)
using *KBasic2-4* **apply** *assumption*
apply (*PLM-subst-method* $(\neg\Box((\neg\varphi) \vee \Box(\neg\psi)))\ (\Diamond(\neg((\neg\varphi) \vee$
$(\Box(\neg\psi)))))$)
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* $\neg((\neg\varphi) \vee (\neg\Diamond\psi))\ \varphi\ \&\ \Diamond\psi$)
using *oth-class-taut-6-a*[*equiv-sym*] **apply** *assumption*
apply (*PLM-subst-method* $\neg((\neg(\Diamond\varphi)) \vee (\neg\Diamond\psi))\ \Diamond\varphi\ \&\ \Diamond\psi$)
using *oth-class-taut-6-a*[*equiv-sym*] **apply** *assumption*
using *2* **by** *assumption*
qed

lemma *S5Basic-13*[*PLM*]:
[$\Diamond(\varphi\ \&\ (\Box\psi)) \equiv (\Diamond\varphi\ \&\ (\Box\psi))$ *in v*]
apply (*PLM-subst-method* $\Diamond\Box\psi\ \Box\psi$)
using *S5Basic-2*[*equiv-sym*] **apply** *assumption*
using *S5Basic-12* **by** *simp*

lemma *S5Basic-14*[*PLM*]:
[$\Box(\varphi \rightarrow (\Box\psi)) \equiv \Box(\Diamond\varphi \rightarrow \psi)$ *in v*]
proof (*rule* $\equiv$*I*; *rule CP*)
assume [$\Box(\varphi \rightarrow \Box\psi)$ *in v*]
moreover {
have $\bigwedge v.$[$\Box(\varphi \rightarrow \Box\psi) \rightarrow (\Diamond\varphi \rightarrow \psi)$ *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 [□(□(φ → □ψ) → (◇φ → ψ)) in v]
        by (rule RN)
      hence [□(□(φ → □ψ)) → □((◇φ → ψ)) in v]
        using qml-1[axiom-instance, deduction] by auto
    }
    ultimately show [□(◇φ → ψ) in v]
      using S5Basic-6 CP vdash-properties-10 by meson
  next
    assume [□(◇φ → ψ) in v]
    moreover {
      fix v
      {
        assume [□(◇φ → ψ) in v]
        hence 1: [□◇φ → □ψ in v]
          using qml-1[axiom-instance, deduction] by auto
        assume [φ in v]
        hence [□◇φ in v]
          using S5Basic-4[deduction] by auto
        hence [□ψ in v]
          using 1[deduction] by auto
      }
      hence [□(◇φ → ψ) in v] ⟹ [φ → □ψ in v]
        using CP by auto
    }
    ultimately show [□(φ → □ψ) in v]
      using S5Basic-6 RN-2 vdash-properties-10 by blast
  qed

lemma sc-eq-box-box-1[PLM]:
  [□(φ → □φ) → (◇φ ≡ □φ) in v]
  proof(rule CP)
    assume 1: [□(φ → □φ) in v]
    hence [□(◇φ → φ) in v]
      using S5Basic-14[equiv-lr] by auto
    hence [◇φ → φ in v]
      using qml-2[axiom-instance, deduction] by auto
    moreover from 1 have [φ → □φ in v]
      using qml-2[axiom-instance, deduction] by auto
    ultimately have [◇φ → □φ in v]
      using ded-thm-cor-3 by auto
    moreover have [□φ → ◇φ in v]
      using qml-2[axiom-instance] T◇
      by (rule ded-thm-cor-3)
    ultimately show [◇φ ≡ □φ in v]
      by (rule ≡I)
  qed

lemma sc-eq-box-box-2[PLM]:
```

[□(φ → □φ) → ((¬□φ) ≡ (□(¬φ))) *in v*]
proof (*rule CP*)
assume [□(φ → □φ) *in v*]
hence [(¬□(¬φ)) ≡ □φ *in v*]
using *sc-eq-box-box-1*[*deduction*] **unfolding** *diamond-def* **by** *auto*
thus [((¬□φ) ≡ (□(¬φ))) *in v*]
by (*meson CP* ≡*I* ≡*E*(*3*)
≡*E*(*4*) ¬¬*I* ¬¬*E*)
qed

lemma *sc-eq-box-box-3*[*PLM*]:
[(□(φ → □φ) & □(ψ → □ψ)) → ((□φ ≡ □ψ) → □(φ ≡ ψ)) *in v*]
proof (*rule CP*)
assume *1*: [(□(φ → □φ) & □(ψ → □ψ)) *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 $\bigwedge v.$ [χ *in v*] $\Longrightarrow$ [$\varphi \rightarrow \Box\psi$ *in v*]
  shows [$\Box\chi$ *in v*] $\Longrightarrow$ [$\Diamond\varphi \rightarrow \psi$ *in v*]
  proof −
    have [$\Box\chi$ *in v*] $\Longrightarrow$ [$\Diamond\varphi \rightarrow \Diamond\Box\psi$ *in v*]
      using *RM-2-b assms* **by** *metis*
    thus [$\Box\chi$ *in v*] $\Longrightarrow$ [$\Diamond\varphi \rightarrow \psi$ *in v*]
      using *B*$\Diamond$ *vdash-properties-10 CP* **by** *metis*
  qed

lemma *derived-S5-rules-2-b*[*PLM*]:
  assumes $\bigwedge v.$ [$\varphi \rightarrow \Box\psi$ *in v*]
  shows [$\Diamond\varphi \rightarrow \psi$ *in v*]
  using *assms derived-S5-rules-2-a all-self-eq-1* **by** *blast*

lemma *BFs-1*[*PLM*]: [$(\forall\,\alpha.\ \Box(\varphi\ \alpha)) \rightarrow \Box(\forall\,\alpha.\ \varphi\ \alpha)$ *in v*]
  proof (*rule derived-S5-rules-1-b*)
    fix *v*
    {
      fix α
      have $\bigwedge v.$[$(\forall\,\alpha\ .\ \Box(\varphi\ \alpha)) \rightarrow \Box(\varphi\ \alpha)$ *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*]:

```
  [◇(∃ α. φ α) → (∃ α . ◇(φ α)) in v]
  proof –
    have [(∀ α. □(¬(φ α))) → □(∀ α. ¬(φ α)) in v]
      using BF by metis
    hence 1: [(¬(□(∀ α. ¬(φ α)))) → (¬(∀ α. □(¬(φ α)))) in v]
      using contraposition-1 by simp
    have 2: [◇(¬(∀ α. ¬(φ α))) → (¬(∀ α. □(¬(φ α)))) in v]
      apply (PLM-subst-method ¬□(∀ α . ¬(φ α)) ◇(¬(∀ α. ¬(φ α))))
      using KBasic2-2 1 by simp+
    have [◇(¬(∀ α. ¬(φ α))) → (∃ α . ¬(□(¬(φ α)))) in v]
      apply (PLM-subst-method ¬(∀ α. □(¬(φ α))) ∃ α. ¬(□(¬(φ α))))
       using cqt-further-2 apply metis
      using 2 by metis
    thus ?thesis
      unfolding exists-def diamond-def by auto
  qed
lemmas BF◇ = BFs-3

lemma BFs-4[PLM]:
  [(∃ α . ◇(φ α)) → ◇(∃ α. φ α) in v]
  proof –
    have 1: [□(∀ α . ¬(φ α)) → (∀ α. □(¬(φ α))) in v]
      using CBF by auto
    have 2: [(∃ α . (¬(□(¬(φ α))))) → (¬(□(∀ α. ¬(φ α)))) in v]
       apply (PLM-subst-method ¬(∀ α. □(¬(φ α))) (∃ α . (¬(□(¬(φ
α))))))
       using cqt-further-2 apply assumption
      using 1 using contraposition-1 by metis
    have [(∃ α . (¬(□(¬(φ α))))) → ◇(¬(∀ α . ¬(φ α))) in v]
     apply (PLM-subst-method ¬(□(∀ α. ¬(φ α))) ◇(¬(∀ α. ¬(φ α))))
       using KBasic2-2 apply assumption
      using 2 by assumption
    thus ?thesis
      unfolding diamond-def exists-def by auto
  qed
lemmas CBF◇ = BFs-4

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

lemma *sign-S5-thm-2*[*PLM*]:
  [$\Diamond(\forall\ \alpha\ .\ \varphi\ \alpha) \rightarrow (\forall\ \alpha\ .\ \Diamond(\varphi\ \alpha))$ *in v*]
  proof −
    {
      fix α
      {
        fix *v*
        have [$(\forall\ \alpha\ .\ \varphi\ \alpha) \rightarrow \varphi\ \alpha$ *in v*]
          using *cqt-orig-1* **by** *metis*
      }
      hence [$\Diamond(\forall\ \alpha\ .\ \varphi\ \alpha) \rightarrow \Diamond(\varphi\ \alpha)$ *in v*]
        using *RM-2* **by** *metis*
    }
    hence [$\forall\ \alpha\ .\ \Diamond(\forall\ \alpha\ .\ \varphi\ \alpha) \rightarrow \Diamond(\varphi\ \alpha)$ *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*]:
  [$\Diamond(\exists\ \alpha\ .\ \varphi\ \alpha\ \&\ \psi\ \alpha) \rightarrow \Diamond((\exists\ \alpha\ .\ \varphi\ \alpha)\ \&\ (\exists\ \alpha\ .\ \psi\ \alpha))$ *in v*]
  by (*simp only*: *RM-2 cqt-further-5*)

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

lemma *sign-S5-thm-5*[*PLM*]:
  [$((\Box(\forall\alpha.\ \varphi\ \alpha \equiv \psi\ \alpha))\ \&\ (\Box(\forall\alpha.\ \psi\ \alpha \equiv \chi\ \alpha))) \rightarrow (\Box(\forall\alpha.\ \varphi\ \alpha \equiv \chi$
$\alpha))$ *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 $[\Box(\forall\,\alpha.\ \varphi\ \alpha \equiv \chi\ \alpha)\ \mathit{in}\ v]$
using *RN-2* **by** *blast*
qed

lemma *id-nec2-1*[*PLM*]:
$[\Diamond((\alpha{::}'a{::}\mathit{id\text{-}eq}) = \beta) \equiv (\alpha = \beta)\ \mathit{in}\ v]$
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*:
$[(\Diamond\varphi) \equiv \psi\ \mathit{in}\ v] \Longrightarrow [(\neg\psi) \equiv \Box(\neg\varphi)\ \mathit{in}\ v]$
proof −
assume $[(\Diamond\varphi) \equiv \psi\ \mathit{in}\ v]$
moreover have $\bigwedge\varphi\ \psi.\ [(\neg\varphi) \equiv \psi\ \mathit{in}\ v] \Longrightarrow [(\neg\psi) \equiv \varphi\ \mathit{in}\ v]$
by *PLM-solver*
ultimately show *?thesis*
unfolding *diamond-def* **by** *blast*
qed

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

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

lemma *exists-desc-box-1*[*PLM*]:
$[(\exists\ \ y\ .\ (y^P) = (\iota x.\ \varphi\ x)) \to (\exists\ \ y\ .\ \Box((y^P) = (\iota x.\ \varphi\ x)))\ \mathit{in}\ v]$
proof (*rule CP*)
assume $[\exists\, y.\ (y^P) = (\iota x.\ \varphi\ x)\ \mathit{in}\ v]$
then obtain y **where** $[(y^P) = (\iota x.\ \varphi\ x)\ \mathit{in}\ v]$
by (*rule* $\exists\,E$)
hence $[\Box(y^P = (\iota x.\ \varphi\ x))\ \mathit{in}\ v]$
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 $[\exists\, y.\ \Box((y^P) = (\iota x.\ \varphi\ x))\ \mathit{in}\ v]$
by (*rule* $\exists\,I$)
qed

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

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

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** *assumption*
apply (*PLM-subst-method* $\{\!|x,F|\!\}$ $\Box\{\!|x,F|\!\}$)
using *en-eq-2* **apply** *assumption*
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 *IsPropositionalInX* φ
and *IsPropositionalInX* 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 *IsPropositionalInXY* φ
and *IsPropositionalInXY* 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 *IsPropositionalInXYZ* φ
and *IsPropositionalInXYZ* ψ
and $\bigwedge x\ y\ z.[\varphi\ (x^P)\ (y^P)\ (z^P) \equiv \psi\ (x^P)\ (y^P)\ (z^P)\ in\ v]$
shows $[(\!|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|\!)$
$\equiv (\!|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \psi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|\!)\ in\ v]$
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 *IsPropositionalInX* φ
and *IsPropositionalInX* ψ
shows $[(\Box(\forall\ x\ .\ \varphi\ (x^P) \equiv \psi\ (x^P))) \rightarrow$
$(\Box(\forall\ x\ .\ (\!|\boldsymbol{\lambda}\ y.\ \varphi\ (y^P),\ x^P|\!) \equiv (\!|\boldsymbol{\lambda}\ y.\ \psi\ (y^P),\ x^P|\!)))\ in\ v]$
apply (*rule qml-1*[*axiom-instance*, *deduction*])
apply (*rule RN*)
proof (*rule CP*, *rule* $\forall I$)
fix *v x*
assume $[\forall x.\ \varphi\ (x^P) \equiv \psi\ (x^P)\ in\ v]$
hence $\bigwedge x.[\varphi\ (x^P) \equiv \psi\ (x^P)\ in\ v]$
by *PLM-solver*
thus $[(\!|\boldsymbol{\lambda}\ y.\ \varphi\ (y^P),\ x^P|\!) \equiv (\!|\boldsymbol{\lambda}\ y.\ \psi\ (y^P),\ x^P|\!)\ in\ v]$
using *assms beta-equiv-eq-1-1* **by** *auto*
qed

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

lemma *beta-equiv-eq-2-3*[*PLM*]:
assumes *IsPropositionalInXYZ* φ
and *IsPropositionalInXYZ* ψ
shows $[(\Box(\forall\ x\ y\ z\ .\ \varphi\ (x^P)\ (y^P)\ (z^P) \equiv \psi\ (x^P)\ (y^P)\ (z^P))) \rightarrow$
$(\Box(\forall\ x\ y\ z\ .\ (\!|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|\!)$
$\equiv (\!|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \psi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|\!)))\ in\ v]$

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 $[\forall\, x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P) \equiv \psi\ (x^P)\ (y^P)\ (z^P)\ in\ v]$
hence $(\bigwedge x\ y\ z.[\varphi\ (x^P)\ (y^P)\ (z^P) \equiv \psi\ (x^P)\ (y^P)\ (z^P)\ in\ v])$
by (*meson* $\forall E$)
thus $[(\!|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \varphi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|\!)$
$\equiv (\!|\boldsymbol{\lambda}^3\ (\lambda\ x\ y\ z.\ \psi\ (x^P)\ (y^P)\ (z^P)),\ x^P,\ y^P,\ z^P|\!)\ in\ v]$
using *assms beta-equiv-eq-1-3* **by** *auto*
qed

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

lemma *beta-C-meta-2*[*PLM*]:
assumes *IsPropositionalInXY* 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 *IsPropositionalInXYZ* 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 *IsPropositionalInX* φ
shows $[\exists\ F.\ \Box(\forall\ x.\ (\!|F,x^P|\!) \equiv \varphi\ (x^P))\ in\ v]$
using *assms* **apply** *cut-tac* **by** *PLM-solver*

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

lemma *relations-3*[*PLM*]:
assumes *IsPropositionalInXYZ* 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** *cut-tac* **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 $[\forall x.\ \Box(\{\!|x^P,F|\!\} \equiv \{\!|x^P,G|\!\})\ \mathit{in}\ v]$
by (*rule* $\forall I$)
thus $[F = G\ \mathit{in}\ v]$
unfolding *identity-defs*
by (*rule BF*[*deduction*])
qed

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

lemma *propositions-lemma-2*[*PLM*]:
$[\boldsymbol{\lambda}^0\ \varphi \equiv \varphi\ \mathit{in}\ v]$
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 $\bigwedge x.[\boldsymbol{\mathcal{A}}(\varphi\ x \equiv \psi\ x)\ \mathit{in}\ v]$
shows $[(\chi{::}\kappa{\Rightarrow}\mathrm{o})\ (\boldsymbol{\iota} x.\ \varphi\ x) = \chi\ (\boldsymbol{\iota} x.\ \psi\ x)\ \mathit{in}\ v]$
proof –
have $[\boldsymbol{\lambda}^0\ (\chi\ (\boldsymbol{\iota} x.\ \varphi\ x)) = \boldsymbol{\lambda}^0\ (\chi\ (\boldsymbol{\iota} x.\ \psi\ x))\ \mathit{in}\ v]$
using *assms lambda-predicates-4-0*
by *blast*
hence $[(\chi\ (\boldsymbol{\iota} x.\ \varphi\ x)) = \boldsymbol{\lambda}^0\ (\chi\ (\boldsymbol{\iota} x.\ \psi\ x))\ \mathit{in}\ v]$
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

TODO 1. *Remark 132?*

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

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

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

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

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

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

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

lemma *thm-relation-negation-2-3*[*PLM*]:
  $[(\neg\llparenthesis F^-, x^P, y^P, z^P\rrparenthesis) \equiv \llparenthesis F, x^P, y^P, z^P\rrparenthesis \ in\ v]$
  using *thm-relation-negation-1-3*[*THEN oth-class-taut-5-d*[*equiv-lr*]]
  apply *cut-tac* **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 *cut-tac* **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** $\varphi{=}\lambda\ G\ .\ \llparenthesis G{,}x^P\rrparenthesis \equiv \llparenthesis F^-{,}x^P\rrparenthesis$, *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** $\varphi{=}\lambda\ G\ .\ \llparenthesis G{,}x^P{,}y^P\rrparenthesis \equiv \llparenthesis F^-{,}x^P{,}y^P\rrparenthesis$, *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*]:
  [$(p::o) \neq (p^-)$ *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*]:
  [$((p::o)^-) = \neg p$ *in v*]
  unfolding *propnot-defs* **using** *propositions-lemma-1* **by** *simp*

lemma *thm-relation-negation-8*[*PLM*]:
  [$(p::o) \neq \neg p$ *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*]:
  [$((p::o) = q) \rightarrow ((\neg p) = (\neg q))$ *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*]:
  [$((p::o) = q) \rightarrow ((p^-) = (q^-))$ *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 [$\Box(\forall\, x.(|F,x^P|)) \vee \Box(\forall\, x.\neg(|F,x^P|))$ *in v*]
    unfolding *NonContingent-def Necessary-defs Impossible-defs* **.**
   hence [$\Box(\forall\, x.\ \neg(|F^-,x^P|)) \vee \Box(\forall\, x.\ \neg(|F,x^P|))$ *in v*]
    apply *cut-tac*
    apply (*PLM-subst1-method* $\lambda\ x\ .\ (|F,x^P|)\ \lambda\ x\ .\ \neg(|F^-,x^P|)$)

using *thm-relation-negation-2-1*[*equiv-sym*] **by** *auto*
hence $[\Box(\forall\, x.\ \neg(|F^-,x^P|)) \vee \Box(\forall\, x.\ (|F^-,x^P|))\ in\ v]$
apply *cut-tac*
apply (*PLM-subst1-goal-method*
$\lambda\ \varphi\ .\ \Box(\forall\, x.\ \neg(|F^-,x^P|)) \vee \Box(\forall\, x.\ \varphi\ x)\ \lambda\ x\ .\ \neg(|F,x^P|))$
using *thm-relation-negation-1-1*[*equiv-sym*] **by** *auto*
hence $[\Box(\forall\, x.\ (|F^-,x^P|)) \vee \Box(\forall\, x.\ \neg(|F^-,x^P|))\ in\ v]$
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 $[\Box(\forall\, x.\ \neg(|F^-,x^P|)) \vee \Box(\forall\, x.\ (|F^-,x^P|))\ in\ v]$
unfolding *NonContingent-def Necessary-defs Impossible-defs*
by (*rule oth-class-taut-3-e*[*equiv-lr*])
hence $[\Box(\forall\, x.(|F,x^P|)) \vee \Box(\forall\, x.(|F^-,x^P|))\ in\ v]$
apply *cut-tac*
apply (*PLM-subst1-method* $\lambda\ x\ .\ \neg(|F^-,x^P|)\ \lambda\ x\ .\ (|F,x^P|)$)
using *thm-relation-negation-2-1* **by** *auto*
hence $[\Box(\forall\, x.\ (|F,x^P|)) \vee \Box(\forall\, x.\ \neg(|F,x^P|))\ in\ v]$
apply *cut-tac*
apply (*PLM-subst1-method* $\lambda\ x\ .\ (|F^-,x^P|)\ \lambda\ x\ .\ \neg(|F,x^P|)$)
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* $F \equiv \Diamond(\exists\ x\ .\ (|F,x^P|))$ **&** $\Diamond(\exists\ x\ .\ \neg(|F,x^P|))$ *in v*]
proof (*rule* $\equiv I$; *rule CP*)
assume [*Contingent F in v*]
hence $[\neg(\Box(\forall\, x.(|F,x^P|)) \vee \Box(\forall\, x.\neg(|F,x^P|)))\ in\ v]$
unfolding *Contingent-def Necessary-defs Impossible-defs* **.**
hence $[(\neg\Box(\forall\, x.(|F,x^P|)))$ **&** $(\neg\Box(\forall\, x.\neg(|F,x^P|)))\ in\ v]$
by (*rule oth-class-taut-6-d*[*equiv-lr*])
hence $[(\Diamond\neg(\forall\, x.\neg(|F,x^P|)))$ **&** $(\Diamond\neg(\forall\, x.(|F,x^P|)))\ in\ v]$
using *KBasic2-2*[*equiv-lr*] **&***I* **&***E* **by** *meson*
thus $[(\Diamond(\exists\ x.(|F,x^P|)))$ **&** $(\Diamond(\exists\, x.\ \neg(|F,x^P|)))\ in\ v]$
unfolding *exists-def* **apply** *cut-tac*
apply (*PLM-subst1-method* $\lambda\ x\ .\ (|F,x^P|)\ \lambda\ x\ .\ \neg\neg(|F,x^P|)$)
using *oth-class-taut-4-b* **by** *auto*
next
assume $[(\Diamond(\exists\ x.(|F,x^P|)))$ **&** $(\Diamond(\exists\, x.\ \neg(|F,x^P|)))\ in\ v]$
hence $[(\Diamond\neg(\forall\, x.\neg(|F,x^P|)))$ **&** $(\Diamond\neg(\forall\, x.(|F,x^P|)))\ in\ v]$
unfolding *exists-def* **apply** *cut-tac*
apply (*PLM-subst1-goal-method*
$\lambda\ \varphi\ .\ (\Diamond\neg(\forall\, x.\neg(|F,x^P|)))$ **&** $(\Diamond\neg(\forall\, x.\ \varphi\ x))\ \lambda\ x\ .\ \neg\neg(|F,x^P|)$)
using *oth-class-taut-4-b*[*equiv-sym*] **by** *auto*
hence $[(\neg\Box(\forall\, x.(|F,x^P|)))$ **&** $(\neg\Box(\forall\, x.\neg(|F,x^P|)))\ in\ v]$
using *KBasic2-2*[*equiv-rl*] **&***I* **&***E* **by** *meson*
hence $[\neg(\Box(\forall\, x.(|F,x^P|)) \vee \Box(\forall\, x.\neg(|F,x^P|)))\ in\ v]$
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*]:
$[\Diamond(\exists\ x\ .\ (\!|F,x^P|\!)\ \&\ (\Diamond(\neg(\!|F,x^P|\!)))) \equiv \Diamond(\exists\ x\ .\ ((\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!)))$*in v*]
proof –
have $[\Diamond(\exists\ x\ .\ (\!|F,x^P|\!)\ \&\ (\Diamond(\neg(\!|F,x^P|\!))))$ *in v*]
$= [(\exists\ x\ .\ \Diamond((\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!))))$ *in v*]
using *BF*$\Diamond$[*deduction*] *CBF*$\Diamond$[*deduction*] **by** *fast*
also have ... $= [\exists\ x\ .\ (\Diamond(\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!)))$ *in v*]
apply (*PLM-subst1-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 ... $= [\exists\ x\ .\ \Diamond(\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!)$ *in v*]
apply (*PLM-subst1-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 ... $= [\exists\ x\ .\ \Diamond((\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!))$ *in v*]
apply (*PLM-subst1-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 ... $= [\Diamond\ (\exists\ x\ .\ ((\neg(\!|F,x^P|\!))\ \&\ \Diamond(\!|F,x^P|\!)))$ *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*]:
$[\Diamond(\exists\ x\ .\ (\!|F,x^P|\!)\ \&\ \Diamond(\neg(\!|F,x^P|\!))) \equiv \Diamond(\exists\ x\ .\ (\!|F^-,x^P|\!)\ \&\ \Diamond(\neg(\!|F^-,x^P|\!)))$
in v]
apply (*PLM-subst1-method* $\lambda\ x\ .\ (\!|F,x^P|\!)\ \lambda\ x\ .\ \neg(\!|F^-,x^P|\!))$
using *thm-relation-negation-2-1*[*equiv-sym*] **apply** *simp*
apply (*PLM-subst1-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*]:
$[\Diamond(\exists\ x\ .\ ((\neg(\!|E!,x^P|\!))\ \&\ (\Diamond(\!|E!,x^P|\!))))$ *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* $\forall I$)
apply (*rule lambda-predicates-2-1*[*axiom-instance*, *equiv-rl*])
apply (*rule IsPropositional-intros*)+
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* $\forall I$)
apply (*rule thm-relation-negation-2-1*[*equiv-rl*])
apply (*rule lambda-predicates-2-1*[*axiom-instance*, *equiv-rl*])
apply (*rule IsPropositional-intros*)+
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* $\vee I(1)$)

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

lemma *thm-noncont-e-e-5*[*PLM*]:
[$\exists$ $(F::\Pi_1)$ *G* . $F \neq G$ **&** *NonContingent F* **&** *NonContingent G in*
v]
apply (*rule-tac* $\alpha{=}L$ **in** $\exists I$, *rule-tac* $\alpha{=}L^-$ **in** $\exists I$)
using $\exists I$ *thm-relation-negation-5-1 thm-noncont-e-e-3*
thm-noncont-e-e-4 **&***I*
by *simp*

lemma *four-distinct-1*[*PLM*]:
[*NonContingent* $(F::\Pi_1) \rightarrow \neg(\exists\ G\ .$ (*Contingent G* **&** $G = F$)) *in* *v*]
proof (*rule CP*)
assume [*NonContingent F in v*]

hence $[\neg(\mathit{Contingent}\ F)\ \mathit{in}\ v]$
unfolding *NonContingent-def Contingent-def*
apply *cut-tac* **by** *PLM-solver*
moreover {
assume $[\exists\ G\ .\ \mathit{Contingent}\ G\ \&\ G = F\ \mathit{in}\ v]$
then obtain P **where** $[\mathit{Contingent}\ P\ \&\ P = F\ \mathit{in}\ v]$
by $(\mathit{rule}\ \exists E)$
hence $[\mathit{Contingent}\ F\ \mathit{in}\ v]$
using **&***E l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *blast*
}
ultimately show $[\neg(\exists G.\ \mathit{Contingent}\ G\ \&\ G = F)\ \mathit{in}\ v]$
using *modus-tollens-1 CP* **by** *blast*
qed

lemma *four-distinct-2*[*PLM*]:
$[\mathit{Contingent}\ (F{::}\Pi_1) \rightarrow \neg(\exists\ G\ .\ (\mathit{NonContingent}\ G\ \&\ G = F))\ \mathit{in}\ v]$
proof (*rule CP*)
assume $[\mathit{Contingent}\ F\ \mathit{in}\ v]$
hence $[\neg(\mathit{NonContingent}\ F)\ \mathit{in}\ v]$
unfolding *NonContingent-def Contingent-def*
apply *cut-tac* **by** *PLM-solver*
moreover {
assume $[\exists\ G\ .\ \mathit{NonContingent}\ G\ \&\ G = F\ \mathit{in}\ v]$
then obtain P **where** $[\mathit{NonContingent}\ P\ \&\ P = F\ \mathit{in}\ v]$
by $(\mathit{rule}\ \exists E)$
hence $[\mathit{NonContingent}\ F\ \mathit{in}\ v]$
using **&***E l-identity*[*axiom-instance*, *deduction*, *deduction*]
by *blast*
}
ultimately show $[\neg(\exists G.\ \mathit{NonContingent}\ G\ \&\ G = F)\ \mathit{in}\ v]$
using *modus-tollens-1 CP* **by** *blast*
qed

lemma *four-distinct-3*[*PLM*]:
$[L \neq (L^-)\ \&\ L \neq E!\ \&\ L \neq (E!^-)\ \&\ (L^-) \neq E!$
$\&\ (L^-) \neq (E!^-)\ \&\ E! \neq (E!^-)\ \mathit{in}\ v]$
proof (*rule* **&***I*)+
show $[L \neq (L^-)\ \mathit{in}\ v]$
by (*rule thm-relation-negation-5-1*)
next
{
assume $[L = E!\ \mathit{in}\ v]$
hence $[\mathit{NonContingent}\ L\ \&\ L = E!\ \mathit{in}\ v]$
using *thm-noncont-e-e-3* **&***I* **by** *auto*
hence $[\exists\ G\ .\ \mathit{NonContingent}\ G\ \&\ G = E!\ \mathit{in}\ v]$
using *thm-noncont-e-e-3* **&***I* $\exists I$ **by** *fast*
}
thus $[L \neq E!\ \mathit{in}\ v]$
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 $[\exists\ G$. *NonContingent G* **&** $G = (E!^-)$ *in v*$]$
using *thm-noncont-e-e-3* **&***I* $\exists I$ **by** *fast*
}
thus $[L \neq (E!^-)$ *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 $[\exists\ G$. *NonContingent G* **&** $G = E!$ *in v*$]$
using *thm-noncont-e-e-3* **&***I* $\exists I$ **by** *fast*
}
thus $[(L^-) \neq E!$ *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 $[\exists\ G$. *NonContingent G* **&** $G = (E!^-)$ *in v*$]$
using *thm-noncont-e-e-3* **&***I* $\exists I$ **by** *fast*
}
thus $[(L^-) \neq (E!^-)$ *in v*$]$
using *four-distinct-2*[*deduction*, *OF thm-cont-e-3*]
modus-tollens-1 CP
by *blast*
next
show $[E! \neq (E!^-)$ *in v*$]$
by (*rule thm-relation-negation-5-1*)
qed
end

lemma *thm-cont-propos-1*[*PLM*]:
$[$*NonContingent* $(p::o) \equiv$ *NonContingent* (p^-) *in v*$]$
proof (*rule* $\equiv I$; *rule CP*)
assume $[$*NonContingent p in v*$]$
hence $[\Box p \lor \Box\neg p$ *in v*$]$
unfolding *NonContingent-def Necessary-defs Impossible-defs* **.**
hence $[\Box(\neg(p^-)) \lor \Box(\neg p)$ *in v*$]$
apply *cut-tac*
apply (*PLM-subst-method p* $\neg(p^-)$)
using *thm-relation-negation-4*[*equiv-sym*] **by** *auto*
hence $[\Box(\neg(p^-)) \lor \Box(p^-)$ *in v*$]$
apply *cut-tac*
apply (*PLM-subst-goal-method* $\lambda\varphi$. $\Box(\neg(p^-)) \lor \Box(\varphi)$ $\neg p$)
using *thm-relation-negation-3*[*equiv-sym*] **by** *auto*

hence $[\Box(p^-) \lor \Box(\neg(p^-))$ *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 $[\Box(\neg(p^-)) \lor \Box(p^-)$ *in* $v]$
unfolding *NonContingent-def Necessary-defs Impossible-defs*
by (*rule oth-class-taut-3-e*[*equiv-lr*])
hence $[\Box(p) \lor \Box(p^-)$ *in* $v]$
apply *cut-tac*
apply (*PLM-subst-goal-method* $\lambda\varphi\ .\ \Box\varphi \lor \Box(p^-)\ \neg(p^-)$)
using *thm-relation-negation-4* **by** *auto*
hence $[\Box(p) \lor \Box(\neg p)$ *in* $v]$
apply *cut-tac*
apply (*PLM-subst-method* p^- $\neg p$)
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* $p \equiv \Diamond p$ **&** $\Diamond(\neg p)$ *in* v]
proof (*rule* $\equiv I$; *rule CP*)
assume [*Contingent* p *in* v]
hence $[\neg(\Box p \lor \Box(\neg p))$ *in* $v]$
unfolding *Contingent-def Necessary-defs Impossible-defs* **.**
hence $[(\neg\Box p)$ **&** $(\neg\Box(\neg p))$ *in* $v]$
by (*rule oth-class-taut-6-d*[*equiv-lr*])
hence $[(\Diamond\neg(\neg p))$ **&** $(\Diamond\neg p)$ *in* $v]$
using *KBasic2-2*[*equiv-lr*] **&***I* **&***E* **by** *meson*
thus $[(\Diamond p)$ **&** $(\Diamond(\neg p))$ *in* $v]$
apply *cut-tac* **apply** *PLM-solver*
apply (*PLM-subst-method* $\neg\neg p$ p)
using *oth-class-taut-4-b*[*equiv-sym*] **by** *auto*
next
assume $[(\Diamond p)$ **&** $(\Diamond\neg(p))$ *in* $v]$
hence $[(\Diamond\neg(\neg p))$ **&** $(\Diamond\neg(p))$ *in* $v]$
apply *cut-tac* **apply** *PLM-solver*
apply (*PLM-subst-method* p $\neg\neg p$)
using *oth-class-taut-4-b* **by** *auto*
hence $[(\neg\Box p)$ **&** $(\neg\Box(\neg p))$ *in* $v]$
using *KBasic2-2*[*equiv-rl*] **&***I* **&***E* **by** *meson*
hence $[\neg(\Box(p) \lor \Box(\neg p))$ *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* $(p::o) \equiv$ *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 p_0 **where**
$p_0 \equiv \forall\, x.\ (\!|E!,x^P|\!) \rightarrow (\!|E!,x^P|\!)$

lemma *thm-noncont-propos-1*[*PLM*]:
[*Necessary* p_0 *in* v]
unfolding *Necessary-defs* p_0*-def*
apply (*rule RN*, *rule* $\forall\, I$)
using *if-p-then-p* **.**

lemma *thm-noncont-propos-2*[*PLM*]:
[*Impossible* $(p_0{}^-)$ *in* v]
unfolding *Impossible-defs*
apply (*PLM-subst-method* $\neg p_0$ $p_0{}^-$)
using *thm-relation-negation-3*[*equiv-sym*] **apply** *simp*
apply (*PLM-subst-method* p_0 $\neg\neg p_0$)
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* (p_0) *in* v]
unfolding *NonContingent-def* **using** *thm-noncont-propos-1*
by (*rule* $\vee I(1)$)

lemma *thm-noncont-propos-4*[*PLM*]:
[*NonContingent* $(p_0{}^-)$ *in* v]
unfolding *NonContingent-def* **using** *thm-noncont-propos-2*
by (*rule* $\vee I(2)$)

lemma *thm-noncont-propos-5*[*PLM*]:
[$\exists$ $(p{::}\mathrm{o})$ q . $p \neq q$ **&** *NonContingent* p **&** *NonContingent* q *in* v]
apply (*rule-tac* $\alpha{=}p_0$ **in** $\exists\, I$, *rule-tac* $\alpha{=}p_0{}^-$ **in** $\exists\, I$)
using $\exists\, I$ *thm-relation-negation-6 thm-noncont-propos-3*
thm-noncont-propos-4 **&***I* **by** *simp*

private definition q_0 **where**
$q_0 \equiv \exists\ x\ .\ (\!|E!,x^P|\!)$ **&** $\Diamond(\neg(\!|E!,x^P|\!))$

lemma *basic-prop-1*[*PLM*]:
[$\exists$ p . $\Diamond p$ **&** $\Diamond(\neg p)$ *in* v]
apply (*rule-tac* $\alpha{=}q_0$ **in** $\exists\, I$) **unfolding** q_0*-def*
using *qml-4*[*axiom-instance*] **by** *simp*

lemma *basic-prop-2*[*PLM*]:
[*Contingent* q_0 *in* v]
unfolding *Contingent-def Necessary-defs Impossible-defs*
apply (*rule oth-class-taut-6-d*[*equiv-rl*])
apply (*PLM-subst-goal-method* $\lambda\ \varphi\ .\ (\neg\Box(\varphi))$ **&** $\neg\Box\neg q_0$ $\neg\neg q_0$)
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* α=q_0 **in** $\exists I$, *rule-tac* α=$q_0{}^-$ **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::Π_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 *cut-tac* **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 *cut-tac* **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 [p₀ = q₀ in v]
        hence [∃ q . NonContingent q & q = q₀ in v]
          using &I thm-noncont-propos-3 ∃ I[where α=p₀]
          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* $p \rightarrow$ *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 $T\Diamond$[*deduction*] **by** *simp*

lemma *cont-true-cont-3*[*PLM*]:
  [*ContingentlyTrue* $p \equiv$ *ContingentlyFalse* (p^-) *in* v]
  unfolding *ContingentlyTrue-def ContingentlyFalse-def*
  apply (*PLM-subst-method* $\neg p$ p^-)
   using *thm-relation-negation-3*[*equiv-sym*] **apply** *simp*
  apply (*PLM-subst-method* p $\neg\neg p$)
  by *PLM-solver+*

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

lemma *cont-tf-thm-1*[*PLM*]:
  [*ContingentlyTrue* $q_0 \vee$ *ContingentlyFalse* q_0 *in* v]
  proof −
    have [$q_0 \vee \neg q_0$ *in* v]
      by *PLM-solver*
    moreover {
      assume [q_0 *in* v]
      hence [q_0 **&** $\Diamond\neg q_0$ *in* v]
        unfolding q_0*-def*
        using *qml-4*[*axiom-instance*,*conj2*] **&**I
        by *auto*
    }
    moreover {
      assume [$\neg q_0$ *in* v]
      hence [($\neg q_0$) **&** $\Diamond q_0$ *in* v]
        unfolding q_0*-def*
        using *qml-4*[*axiom-instance*,*conj1*] **&**I
        by *auto*
    }
    ultimately show *?thesis*
      unfolding *ContingentlyTrue-def ContingentlyFalse-def*
      using $\vee E(4)$ *CP* **by** *auto*
  qed

lemma *cont-tf-thm-2*[*PLM*]:
  [*ContingentlyFalse* $q_0 \vee$ *ContingentlyFalse* (${q_0}^-$) *in* v]
  using *cont-tf-thm-1 cont-true-cont-3*[**where** p=q_0]
        cont-true-cont-4[**where** p=q_0]

apply *cut-tac* **by** *PLM-solver*

lemma *cont-tf-thm-3*[*PLM*]:
[∃ *p* . *ContingentlyTrue p in v*]
proof (*rule* ∨*E*(*1*); (*rule CP*)?)
show [*ContingentlyTrue* q_0 ∨ *ContingentlyFalse* q_0 *in v*]
using *cont-tf-thm-1* **.**
next
assume [*ContingentlyTrue* q_0 *in v*]
thus *?thesis*
using ∃ *I* **by** *metis*
next
assume [*ContingentlyFalse* q_0 *in v*]
hence [*ContingentlyTrue* (q_0^-) *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* q_0 ∨ *ContingentlyFalse* q_0 *in v*]
using *cont-tf-thm-1* **.**
next
assume [*ContingentlyTrue* q_0 *in v*]
hence [*ContingentlyFalse* (q_0^-) *in v*]
using *cont-true-cont-3*[*equiv-lr*] **by** *simp*
thus *?thesis*
using ∃ *I* **by** *metis*
next
assume [*ContingentlyFalse* q_0 *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 *cut-tac* **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 cut-tac 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 [⦇(λx. ◇⦇E!,x^P⦈), x^P⦈ ≡ ◇⦇E!,x^P⦈ in v]
        apply (rule beta-C-meta-1) by (rule IsPropositional-intros)+
      ultimately have [⦇(λx. ¬◇⦇E!,x^P⦈), x^P⦈ ≡ ◇⦇E!,x^P⦈ in v]
        using l-identity[axiom-instance, deduction, deduction] by fast
      moreover have [⦇(λx. ¬◇⦇E!,x^P⦈), x^P⦈ ≡ ¬◇⦇E!,x^P⦈ in v]
        apply (rule beta-C-meta-1) by (rule IsPropositional-intros)+
      ultimately have [◇⦇E!,x^P⦈ ≡ ¬◇⦇E!,x^P⦈ in v]
        apply cut-tac by PLM-solver
    }
    thus ?thesis
      using oth-class-taut-1-b modus-tollens-1 CP
      by blast
  qed

lemma oa-contingent-2[PLM]:
  [⦇O!,x^P⦈ ≡ ¬⦇A!,x^P⦈ in v]
  proof −
      have [⦇(λx. ¬◇⦇E!,x^P⦈), x^P⦈ ≡ ¬◇⦇E!,x^P⦈ in v]
        apply (rule beta-C-meta-1)
        by (rule IsPropositional-intros)+
```

```
      hence [(¬(|(λx. ¬◇(|E!,x^P|)), x^P|)) ≡ ◇(|E!,x^P|) in v]
        using oth-class-taut-5-d[equiv-lr] oth-class-taut-4-b[equiv-sym]
              ≡E(5) by blast
      moreover have [(|(λx. ◇(|E!,x^P|)), x^P|) ≡ ◇(|E!,x^P|) in v]
        apply (rule beta-C-meta-1)
        by (rule IsPropositional-intros)+
      ultimately show ?thesis
        unfolding Ordinary-def Abstract-def
        apply cut-tac by PLM-solver
    qed

  lemma oa-contingent-3[PLM]:
    [(|A!,x^P|) ≡ ¬(|O!,x^P|) in v]
    using oa-contingent-2
    apply cut-tac 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-subst1-method λ x . ◇(|E!,x^P|) λ x . (|λx. ◇(|E!,x^P|),x^P|))
      apply (rule beta-C-meta-1[equiv-sym]; (rule IsPropositional-intros)+)
     using BF◇[deduction, OF thm-cont-prop-2[equiv-lr, OF thm-cont-e-2,
conj1]]
      by (rule T◇[deduction])
    subgoal
      apply (PLM-subst1-method λ x . (|A!,x^P|) λ x . ¬(|O!,x^P|))
       using oa-contingent-3 apply simp
      using cqt-further-5[deduction,conj1, OF A-objects[axiom-instance]]
      by (rule T◇[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◇[deduction])
    subgoal
      unfolding Abstract-def
     apply (PLM-subst1-method λ x . ¬◇(|E!,x^P|) λ x . (|λx. ¬◇(|E!,x^P|),x^P|))
      apply (rule beta-C-meta-1[equiv-sym]; (rule IsPropositional-intros)+)
      apply (PLM-subst1-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 (rule IsPropositional-intros)+
    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 cut-tac
      apply (PLM-subst-method ⦇λx. ¬⦇A!,x^P⦈,x^P⦈ (¬⦇A!,x^P⦈))
       apply (rule beta-C-meta-1; (rule IsPropositional-intros)+)
      by assumption
    hence [⦇O!,x^P⦈ ≡ ¬⦇O!,x^P⦈ in v]
      using oa-contingent-2 apply cut-tac 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 (rule beta-C-meta-1[equiv-sym];
              (rule IsPropositional-intros)+)
      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 (rule IsPropositional-intros)+
    ultimately show ?thesis
      unfolding propnot-defs
      using oa-contingent-3
      apply cut-tac 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 [⦇O!,x^P⦈ in v]
    hence [◇⦇E!,x^P⦈ in v]
      unfolding Ordinary-def apply cut-tac
      apply (rule beta-C-meta-1[equiv-lr])
```

by (*rule IsPropositional-intros* | *assumption*)+
hence $[\Box\Diamond\llparenthesis E!,x^P\rrparenthesis$ *in v*]
using *qml-3*[*axiom-instance*, *deduction*] **by** *auto*
thus $[\Box\llparenthesis O!,x^P\rrparenthesis$ *in v*]
unfolding *Ordinary-def*
apply *cut-tac*
apply (*PLM-subst-method* $\Diamond\llparenthesis E!,x^P\rrparenthesis$ $\llparenthesis\boldsymbol{\lambda}x.\ \Diamond\llparenthesis E!,x^P\rrparenthesis,x^P\rrparenthesis$)
by (*rule beta-C-meta-1*[*equiv-sym*],
(*rule IsPropositional-intros* | *assumption*)+)
qed

lemma *oa-facts-2*[*PLM*]:
$[\llparenthesis A!,x^P\rrparenthesis \rightarrow \Box\llparenthesis A!,x^P\rrparenthesis$ *in v*]
proof (*rule CP*)
assume $[\llparenthesis A!,x^P\rrparenthesis$ *in v*]
hence $[\neg\Diamond\llparenthesis E!,x^P\rrparenthesis$ *in v*]
unfolding *Abstract-def* **apply** *cut-tac*
apply (*rule beta-C-meta-1*[*equiv-lr*])
by (*rule IsPropositional-intros* | *assumption*)+
hence $[\Box\Box\neg\llparenthesis E!,x^P\rrparenthesis$ *in v*]
using *KBasic2-4*[*equiv-rl*] *4*$\Box$[*deduction*] **by** *auto*
hence $[\Box\neg\Diamond\llparenthesis E!,x^P\rrparenthesis$ *in v*]
apply *cut-tac*
apply (*PLM-subst-method* $\Box\neg\llparenthesis E!,x^P\rrparenthesis$ $\neg\Diamond\llparenthesis E!,x^P\rrparenthesis$)
using *KBasic2-4* **by** *auto*
thus $[\Box\llparenthesis A!,x^P\rrparenthesis$ *in v*]
unfolding *Abstract-def*
apply *cut-tac*
apply (*PLM-subst-method* $\neg\Diamond\llparenthesis E!,x^P\rrparenthesis$ $\llparenthesis\boldsymbol{\lambda}x.\ \neg\Diamond\llparenthesis E!,x^P\rrparenthesis,x^P\rrparenthesis$)
by (*rule beta-C-meta-1*[*equiv-sym*], (*rule IsPropositional-intros* | *assumption*)+)
qed

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

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

lemma *oa-facts-5*[*PLM*]:
$[\Diamond\llparenthesis O!,x^P\rrparenthesis \equiv \Box\llparenthesis O!,x^P\rrparenthesis$ *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*]:
$[\Diamond\llparenthesis A!,x^P\rrparenthesis \equiv \Box\llparenthesis A!,x^P\rrparenthesis$ *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*]:

$[(|O!,x^P|) \equiv \mathcal{A}(|O!,x^P|)$ *in* $v]$
apply (*rule* ≡*I*; *rule CP*)
apply (*rule nec-imp-act*[*deduction*, *OF oa-facts-1*[*deduction*]]; *assumption*)
proof −
assume $[\mathcal{A}(|O!,x^P|)$ *in* $v]$
hence $[\mathcal{A}(\Diamond(|E!,x^P|))$ *in* $v]$
unfolding *Ordinary-def* **apply** *cut-tac*
apply (*PLM-subst-method* $(|\boldsymbol{\lambda}x.\ \Diamond(|E!,x^P|),x^P|)\ \Diamond(|E!,x^P|)$)
by (*rule beta-C-meta-1*, (*rule IsPropositional-intros* | *assumption*)+)
hence $[\Diamond(|E!,x^P|)$ *in* $v]$
using *Act-Basic-6*[*equiv-rl*] **by** *auto*
thus $[(|O!,x^P|)$ *in* $v]$
unfolding *Ordinary-def* **apply** *cut-tac*
apply (*PLM-subst-method* $\Diamond(|E!,x^P|)\ (|\boldsymbol{\lambda}x.\ \Diamond(|E!,x^P|),x^P|)$)
by (*rule beta-C-meta-1*[*equiv-sym*],
(*rule IsPropositional-intros* | *assumption*)+)
qed

lemma *oa-facts-8*[*PLM*]:
$[(|A!,x^P|) \equiv \mathcal{A}(|A!,x^P|)$ *in* $v]$
apply (*rule* ≡*I*; *rule CP*)
apply (*rule nec-imp-act*[*deduction*, *OF oa-facts-2*[*deduction*]]; *assumption*)
proof −
assume $[\mathcal{A}(|A!,x^P|)$ *in* $v]$
hence $[\mathcal{A}(\neg\Diamond(|E!,x^P|))$ *in* $v]$
unfolding *Abstract-def* **apply** *cut-tac*
apply (*PLM-subst-method* $(|\boldsymbol{\lambda}x.\ \neg\Diamond(|E!,x^P|),x^P|)\ \neg\Diamond(|E!,x^P|)$)
by (*rule beta-C-meta-1*, (*rule IsPropositional-intros* | *assumption*)+)
hence $[\mathcal{A}(\Box\neg(|E!,x^P|))$ *in* $v]$
apply *cut-tac*
apply (*PLM-subst-method* $(\neg\Diamond(|E!,x^P|))\ (\Box\neg(|E!,x^P|))$)
using *KBasic2-4*[*equiv-sym*] **by** *auto*
hence $[\neg\Diamond(|E!,x^P|)$ *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** *cut-tac*
apply (*PLM-subst-method* $\neg\Diamond(|E!,x^P|)\ (|\boldsymbol{\lambda}x.\ \neg\Diamond(|E!,x^P|),x^P|)$)
by (*rule beta-C-meta-1*[*equiv-sym*], (*rule IsPropositional-intros* | *assumption*)+)
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* **&** $(\forall\ x\ .\ (\Diamond(|F,x^P|) \rightarrow \Box(|F,x^P|)))$ *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 cut-tac
            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 cut-tac apply (PLM-subst-method ⦇λx. ¬⦇F,x^P⦈,x^P⦈
¬⦇F,x^P⦈)
               apply (rule beta-C-meta-1; rule IsPropositional-intros)
              by simp
            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 cut-tac
            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 cut-tac 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! ≠ (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! ≠ E! & A! ≠ (E!⁻) & A! ≠ PLM.L & A! ≠ (PLM.L⁻) in v]
  proof ((rule &I)+)
    show [A! ≠ E! in v]
      using cont-nec-fact2-2 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 [A! ≠ (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 $[(\!|O!,x^P|\!)\ in\ v] \land [(\!|O!,y^P|\!)\ in\ v] \land [\Box(\forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!))\ in\ v]$
      using *eq-E-simple-1*[*equiv-lr*] **using** **&**E **by** *blast*
    hence $[\Box(\!|O!,x^P|\!)\ in\ v] \land [\Box(\!|O!,y^P|\!)\ in\ v]$
      $\land [\Box\Box(\forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!))\ in\ v]$
      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 *cut-tac*
      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 ... = [¬⦇(λ² (λ x y . (x^P) =_E (y^P))), x^P, y^P⦈ in v]
      unfolding propnot-defs using beta-C-meta-2[equiv-lr]
      beta-C-meta-2[equiv-rl] IsPropositional-intros by fast
    also have ... = [¬((x^P) =_E (y^P)) in v]
      apply (PLM-subst-method
             ⦇(λ² (λ x y . (x^P) =_E (y^P))), x^P, y^P⦈
             (x^P) =_E (y^P))
       apply (rule beta-C-meta-2) unfolding identity-defs
       apply (rule IsPropositional-intros)
      by auto
    finally show ?thesis
      using ≡I CP by presburger
  qed

lemma id-nec4-1[PLM]:
  [((x^P) ≠_E (y^P)) ≡ □((x^P) ≠_E (y^P)) in v]
  proof –
    have [(¬((x^P) =_E (y^P))) ≡ □(¬((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 cut-tac
      apply (PLM-subst-method (¬((x^P) =_E (y^P))) (x^P) ≠_E (y^P))
      using thm-neg-eqE[equiv-sym] by auto
  qed

lemma id-nec4-2[PLM]:
  [◇((x^P) ≠_E (y^P)) ≡ ((x^P) ≠_E (y^P)) in v]
  using ≡I id-nec4-1[equiv-lr] derived-S5-rules-2-b CP T◇ by simp

lemma id-act-1[PLM]:
  [((x^P) =_E (y^P)) ≡ (𝓐((x^P) =_E (y^P))) in v]
  proof (rule ≡I; rule CP)
    assume [(x^P) =_E (y^P) in v]
    hence [□((x^P) =_E (y^P)) in v]
      using id-nec3-1[equiv-lr] by auto
    thus [𝓐((x^P) =_E (y^P)) in v]
      using nec-imp-act[deduction] by fast
  next
    assume [𝓐((x^P) =_E (y^P)) in v]
    hence [𝓐(⦇O!,x^P⦈ & ⦇O!,y^P⦈ & □(∀ F . ⦇F,x^P⦈ ≡ ⦇F,y^P⦈)) in
v]
      apply cut-tac
      apply (PLM-subst-method
             (x^P) =_E (y^P)
             (⦇O!,x^P⦈ & ⦇O!,y^P⦈ & □(∀ F . ⦇F,x^P⦈ ≡ ⦇F,y^P⦈)))
      using eq-E-simple-1 by auto
    hence [𝓐⦇O!,x^P⦈ & 𝓐⦇O!,y^P⦈ & 𝓐(□(∀ F . ⦇F,x^P⦈ ≡ ⦇F,y^P⦈))
in v]
      using Act-Basic-2[equiv-lr] &I &E by meson
    thus [(x^P) =_E (y^P) in v]
      apply cut-tac 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*]:
$[((x^P) \neq_E (y^P)) \equiv (\boldsymbol{\mathcal{A}}((x^P) \neq_E (y^P))) \ in \ v]$
apply (*PLM-subst-method* $(\neg((x^P) =_E (y^P)))$ $((x^P) \neq_E (y^P)))$
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*: $[\boldsymbol{\mathcal{A}}(\alpha = \beta) \ in \ v] \Longrightarrow [(\alpha = \beta) \ in \ v]$

instantiation ν :: *id-act*
begin
instance proof
interpret *PLM* **.**
fix x::ν **and** y::ν **and** v::i
assume $[\boldsymbol{\mathcal{A}}(x = y) \ in \ v]$
hence $[\boldsymbol{\mathcal{A}}(((x^P) =_E (y^P)) \lor ((\!|A!,x^P|\!) \ \& \ (\!|A!,y^P|\!)$
$\& \ \Box(\forall \ F \ . \ \{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\}))) \ in \ v]$
unfolding *identity-defs* **by** *auto*
hence $[\boldsymbol{\mathcal{A}}(((x^P) =_E (y^P))) \lor \boldsymbol{\mathcal{A}}(((\!|A!,x^P|\!) \ \& \ (\!|A!,y^P|\!)$
$\& \ \Box(\forall \ F \ . \ \{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\}))) \ in \ v]$
using *Act-Basic-10*[*equiv-lr*] **by** *auto*
moreover {
assume $[\boldsymbol{\mathcal{A}}(((x^P) =_E (y^P))) \ in \ v]$
hence $[(x^P) = (y^P) \ in \ v]$
using *id-act-1*[*equiv-rl*] *eq-E-simple-2*[*deduction*] **by** *auto*
}
moreover {
assume $[\boldsymbol{\mathcal{A}}((\!|A!,x^P|\!) \ \& \ (\!|A!,y^P|\!) \ \& \ \Box(\forall \ F \ . \ \{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\}))$
$in \ v]$
hence $[\boldsymbol{\mathcal{A}}(\!|A!,x^P|\!) \ \& \ \boldsymbol{\mathcal{A}}(\!|A!,y^P|\!) \ \& \ \boldsymbol{\mathcal{A}}(\Box(\forall \ F \ . \ \{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\}))$
$in \ v]$
using *Act-Basic-2*[*equiv-lr*] &I &E **by** *meson*
hence $[(\!|A!,x^P|\!) \ \& \ (\!|A!,y^P|\!) \ \& \ (\Box(\forall \ F \ . \ \{\!|x^P,F|\!\} \equiv \{\!|y^P,F|\!\})) \ in \ v]$
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** $\lor I$ **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 Π_1 :: *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 identityₒ-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 blast
      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
```

unfolding *identity*$_3$*-def* **by** *blast*
hence [*?p x y in v*]
using &*I* &*E id-act-prop Act-Basic-2*[*equiv-lr*] **by** *metis*
}
hence [$\forall$ *y* . *?p x y in v*]
by (*rule* $\forall$ *I*)
}
thus [$F = G$ *in v*]
unfolding *identity*$_3$*-def* **by** (*rule* $\forall$ *I*)
qed
end

context *PLM*
begin
lemma *id-act-3*[*PLM*]:
[$((\alpha::('a::id\text{-}act)) = \beta) \equiv \boldsymbol{\mathcal{A}}(\alpha = \beta)$ *in v*]
using $\equiv$*I CP id-nec*[*equiv-lr*, *THEN nec-imp-act*[*deduction*]]
id-act-prop **by** *metis*

lemma *id-act-4*[*PLM*]:
[$((\alpha::('a::id\text{-}act)) \neq \beta) \equiv \boldsymbol{\mathcal{A}}(\alpha \neq \beta)$ *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*]:
[$(y^P) = (\boldsymbol{\iota}x \ . \ x = y)$ *in v*]
using *descriptions*[*axiom-instance*,*equiv-rl*]
id-act-3[*equiv-sym*] $\forall$ *I* **by** *fast*

TODO 2. *More discussion/thought about eta conversion and the strength of the axiom lambda-predicates-3-$*$ which immediately implies the following very general lemmas.*

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⦈ $\vee$ ⦇*A!*,x^P⦈ *in v*]
proof (*rule* $\forall$*I*)
fix *x*
have [$\Diamond$⦇*E!*,x^P⦈ $\vee$ $\neg\Diamond$⦇*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*]
by (*rule beta-C-meta-1[equiv-sym]*; (*rule IsPropositional-intros*)+)
moreover have [($\neg\Diamond$⦇*E!*,x^P⦈) $\equiv$ ⦇$\boldsymbol{\lambda}$ *y* . $\neg\Diamond$⦇*E!*,y^P⦈, x^P⦈ *in v*]
by (*rule beta-C-meta-1[equiv-sym]*; (*rule IsPropositional-intros*)+)
ultimately show [⦇*O!*, x^P⦈ $\vee$ ⦇*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]*:
[⦇*O!*,*x*⦈ $\rightarrow$ ($x =_E x$) *in v*]
proof (*rule CP*)
assume [⦇*O!*,*x*⦈ *in v*]
moreover have [$\Box$($\forall$ *F* . ⦇*F*,*x*⦈ $\equiv$ ⦇*F*,*x*⦈) *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*: [⦇*O!*,*x*⦈ **&** ⦇*O!*,*y*⦈ **&** $\Box$($\forall$ *F* . ⦇*F*,*x*⦈ $\equiv$ ⦇*F*,*y*⦈) *in v*]
using *eq-E-simple-1[equiv-lr]* **by** *simp*
have [$\Box$($\forall$ *F* . ⦇*F*,*y*⦈ $\equiv$ ⦇*F*,*x*⦈) *in v*]
apply (*PLM-subst1-method*
λ *F* . ⦇*F*,*x*⦈ $\equiv$ ⦇*F*,*y*⦈
λ *F* . ⦇*F*,*y*⦈ $\equiv$ ⦇*F*,*x*⦈)

```
      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)) → (x =E z) in v]
  proof (rule CP)
    assume a: [(x =E y) & (y =E z) in v]
    have [□((∀ F . ⦇F,x⦈ ≡ ⦇F,y⦈) & (∀ F . ⦇F,y⦈ ≡ ⦇F,z⦈)) 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 [((∀ F . ⦇F,x⦈ ≡ ⦇F,y⦈) & (∀ F . ⦇F,y⦈ ≡ ⦇F,z⦈))
              → (∀ F . ⦇F,x⦈ ≡ ⦇F,z⦈) in w]
          by PLM-solver
      }
      hence [□(((∀ F . ⦇F,x⦈ ≡ ⦇F,y⦈) & (∀ F . ⦇F,y⦈ ≡ ⦇F,z⦈))
            → (∀ F . ⦇F,x⦈ ≡ ⦇F,z⦈)) in v]
        by (rule RN)
    }
    ultimately have [□(∀ F . ⦇F,x⦈ ≡ ⦇F,z⦈) 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]:
  [(⦇O!,x^P⦈ ∨ ⦇O!,y^P⦈) → ((x^P = y^P) ≡ (x^P =E y^P)) in v]
  proof (rule CP)
    assume [⦇O!,x^P⦈ ∨ ⦇O!,y^P⦈ in v]
    moreover {
      assume [⦇O!,x^P⦈ in v]
      hence [(x^P = y^P) ≡ (x^P =E y^P) in v]
        using ≡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) ≡ (x^P =E y^P) in v]
        using ≡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) ≡ (x^P =E y^P) in v]
      using intro-elim-4-a CP by blast
  qed
```

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

TODO 3. *Check the proof in PM. The last part of the proof by contraposition seems invalid.*

lemma *ord-eq-E2*[*PLM*]:
$[((\llparenthesis O!,x^P \rrparenthesis \ \& \ \llparenthesis O!,y^P \rrparenthesis) \rightarrow$
$((x^P \neq y^P) \equiv (\boldsymbol{\lambda} z \ . \ z^P =_E x^P) \neq (\boldsymbol{\lambda} z \ . \ z^P =_E y^P)) \ in \ v]$
proof (*rule CP*; *rule* $\equiv I$; *rule CP*)
assume *ord-xy*: $[\llparenthesis O!,x^P \rrparenthesis \ \& \ \llparenthesis O!,y^P \rrparenthesis \ in \ v]$
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 $[\llparenthesis \boldsymbol{\lambda} z \ . \ z^P =_E x^P, \ x^P \rrparenthesis \ in \ v]$
apply (*rule beta-C-meta-1*[*equiv-rl*])
unfolding *identity*$_E$*-infix-def*
apply (*rule IsPropositional-intros*)
using *ord-eq-Eequiv-1*[*deduction*] *ord-xy*[*conj1*]
unfolding *identity*$_E$*-infix-def* **by** *presburger*
ultimately have $[\llparenthesis \boldsymbol{\lambda} z \ . \ z^P =_E y^P, \ x^P \rrparenthesis \ in \ v]$
using *l-identity*[*axiom-instance*, *deduction*, *deduction*] **by** *fast*
hence $[x^P =_E y^P \ in \ v]$
using *beta-C-meta-1*[*equiv-lr*] *IsPropositional-intros*
unfolding *identity*$_E$*-infix-def* **by** *fast*
}
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*: $[\llparenthesis O!,x^P \rrparenthesis \ \& \ \llparenthesis O!,y^P \rrparenthesis \ in \ v]$
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 [□⦇O!,x^P⦈ in v]
      using oa-facts-1[deduction] by simp
    moreover hence [□(⦇O!,x^P⦈ → (¬(∃ F . ⦃x^P, F⦄))) in v]
      using nocoder[axiom-necessitation, axiom-instance] by simp
    ultimately show [□(¬(∃ 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-subst1-method  λ x . ◇⦇E!,x^P⦈ λ x . ⦇λy. ◇⦇E!,y^P⦈,
x^P⦈)
   apply (rule beta-C-meta-1[equiv-sym], rule IsPropositional-intros)
  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-subst1-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-subst1-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*]:
$[\Box(\neg(\forall\ x\ .\ (\!|E!,x^P|\!)))\ in\ v]$
proof −
{
fix v
have $[\exists\ x\ .\ (\!|A!,x^P|\!)$ **&** $(\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv (F = F))\ in\ v]$
using *A-objects*[*axiom-instance*] **by** *simp*
hence $[\exists\ x\ .\ (\!|A!,x^P|\!)\ in\ v]$
using *cqt-further-5*[*deduction,conj1*] **by** *fast*
then obtain a **where**
$[(\!|A!,a^P|\!)\ in\ v]$
by (*rule* $\exists$ *E*)
hence $[\neg(\Diamond(\!|E!,a^P|\!))\ in\ v]$
unfolding *Abstract-def*
using *beta-C-meta-1*[*equiv-lr*] *IsPropositional-intros*
by *fast*
hence $[(\neg(\!|E!,a^P|\!))\ in\ v]$
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*]:
$[\exists!\ x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv (F = G))\ in\ v]$
using *A-objects-unique* **.**

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

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

have $[\boldsymbol{\mathcal{A}}(\varphi\ \beta \rightarrow \beta = \alpha)\ in\ v]$
using *2* **by** $(rule\ \forall\ E)$
hence $[\boldsymbol{\mathcal{A}}(\varphi\ \beta) \rightarrow (\beta = \alpha)\ in\ v]$
using *logic-actual-nec-2*[*axiom-instance*, *equiv-lr*, *deduction*]
id-act-3[*equiv-rl*] *CP* **by** *blast*
}
hence $[\forall\ \ \beta\ .\ \boldsymbol{\mathcal{A}}(\varphi\ \beta) \rightarrow (\beta = \alpha)\ in\ v]$
by $(rule\ \forall\ I)$
thus $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}\varphi\ \alpha\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta \rightarrow \beta = \alpha)\ in\ v]$
using *1*[*conj1*] **&**$I\ \exists\ I$ **by** *fast*
next
assume $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}\varphi\ \alpha\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta \rightarrow \beta = \alpha)\ in\ v]$
then obtain α **where** *1*:
$[\boldsymbol{\mathcal{A}}\varphi\ \alpha\ \&\ (\forall\,\beta.\ \boldsymbol{\mathcal{A}}\varphi\ \beta \rightarrow \beta = \alpha)\ in\ v]$
by $(rule\ \exists\ E)$
{
fix β
have $[\boldsymbol{\mathcal{A}}(\varphi\ \beta) \rightarrow \beta = \alpha\ in\ v]$
using *1*[*conj2*] **by** $(rule\ \forall\ E)$
hence $[\boldsymbol{\mathcal{A}}(\varphi\ \beta \rightarrow \beta = \alpha)\ in\ v]$
using *logic-actual-nec-2*[*axiom-instance*, *equiv-rl*] *id-act-3*[*equiv-lr*]
vdash-properties-10 CP **by** *blast*
}
hence $[\forall\ \ \beta\ .\ \boldsymbol{\mathcal{A}}(\varphi\ \beta \rightarrow \beta = \alpha)\ in\ v]$
by $(rule\ \forall\ I)$
hence $[\boldsymbol{\mathcal{A}}(\forall\ \ \beta\ .\ \varphi\ \beta \rightarrow \beta = \alpha)\ in\ v]$
using *logic-actual-nec-3*[*axiom-instance*, *equiv-rl*] **by** *fast*
hence $[\boldsymbol{\mathcal{A}}(\varphi\ \alpha\ \&\ (\forall\ \ \beta\ .\ \varphi\ \beta \rightarrow \beta = \alpha))\ in\ v]$
using *1*[*conj1*] *Act-Basic-2*[*equiv-rl*] **&**I **by** *blast*
hence $[\exists\,\alpha.\ \boldsymbol{\mathcal{A}}(\varphi\ \alpha\ \&\ (\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha))\ in\ v]$
using $\exists\ I$ **by** *fast*
thus $[\boldsymbol{\mathcal{A}}(\exists\,\alpha.\ \varphi\ \alpha\ \&\ (\forall\,\beta.\ \varphi\ \beta \rightarrow \beta = \alpha))\ in\ v]$
using *Act-Basic-11*[*equiv-rl*] **by** *fast*
qed

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

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

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

lemma *can-ab2*[*PLM*]:
$[(y^P = (\boldsymbol{\iota}x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ \ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F))) \rightarrow (\!|A!,y^P|\!)\ in\ v]$
proof $(rule\ CP)$
assume $[y^P = (\boldsymbol{\iota}x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ \ F\ .\ \{\!|x^P,F|\!\} \equiv \varphi\ F))\ in\ v]$

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*]:
$[\{\!| \boldsymbol{\iota}x\ .\ (\!| A!,x^P |\!) \ \&\ (\forall\ F\ .\ \{\!| x^P,F |\!\} \equiv \varphi\ F),\ G |\!\} \equiv \varphi\ G\ in\ dw]$
proof −
obtain a **where**
$[a^P = (\boldsymbol{\iota}x\ .\ (\!| A!,x^P |\!) \ \&\ (\forall\ F\ .\ \{\!| x^P,F |\!\} \equiv \varphi\ F))\ in\ dw]$
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

TODO 4. *Have another look at remark 185.*

notepad
begin
let $?x = \boldsymbol{\iota}x\ .\ (\!| A!,x^P |\!) \ \&\ (\forall\ F\ .\ \{\!| x^P,\ F |\!\} \equiv (\exists\ q\ .\ q\ \&\ F = (\boldsymbol{\lambda}\ y\ .\ q)))$
have $[(\exists\ p\ .\ ContingentlyTrue\ p)\ in\ dw]$
using *cont-tf-thm-3* **by** *auto*
then obtain p_1 **where** $[ContingentlyTrue\ p_1\ in\ dw]$ **by** (*rule* $\exists$ *E*)
hence $[p_1\ in\ dw]$ **unfolding** *ContingentlyTrue-def* **using** &*E* **by** *fast*
hence $[p_1\ \&\ (\boldsymbol{\lambda}\ y\ .\ p_1) = (\boldsymbol{\lambda}\ y\ .\ p_1)\ in\ dw]$ **using** &*I* *id-eq-1* **by** *fast*
hence $[\exists\ q\ .\ q\ \&\ (\boldsymbol{\lambda}\ y\ .\ p_1) = (\boldsymbol{\lambda}\ y\ .\ q)\ in\ dw]$ **using** $\exists$ *I* **by** *fast*
moreover have $[\{\!| ?x,\ \boldsymbol{\lambda}\ y\ .\ p_1 |\!\} \equiv (\exists\ q\ .\ q\ \&\ (\boldsymbol{\lambda}\ y\ .\ p_1) = (\boldsymbol{\lambda}\ y\ .\ q))\ in\ dw]$
using *desc-encode* **by** *fast*
ultimately have $[\{\!| ?x,\ \boldsymbol{\lambda}\ y\ .\ p_1 |\!\}\ in\ dw]$
using $\equiv$*E* **by** *blast*
hence $[\Box\{\!| ?x,\ \boldsymbol{\lambda}\ y\ .\ p_1 |\!\}\ in\ dw]$
using *encoding*[*axiom-instance*,*deduction*] **by** *fast*
hence $\forall\ v\ .\ [\{\!| ?x,\ \boldsymbol{\lambda}\ y\ .\ p_1 |\!\}\ in\ v]$
using *Semantics.T6* **by** *simp*
end

lemma *desc-nec-encode*[*PLM*]:
$[\{\!| \boldsymbol{\iota}x\ .\ (\!| A!,x^P |\!) \ \&\ (\forall\ F\ .\ \{\!| x^P,F |\!\} \equiv \varphi\ F),\ G |\!\} \equiv \boldsymbol{\mathcal{A}}(\varphi\ G)\ in\ v]$
proof −
obtain a **where**
$[a^P = (\boldsymbol{\iota}x\ .\ (\!| A!,x^P |\!) \ \&\ (\forall\ F\ .\ \{\!| x^P,F |\!\} \equiv \varphi\ F))\ in\ v]$
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 $[\boldsymbol{\mathcal{A}}(\{\!|a^P,\ G|\!\} \equiv \varphi\ G)\ \mathit{in}\ v]$
using $\forall E$ **by** *fast*
hence $[\boldsymbol{\mathcal{A}}\{\!|a^P,\ G|\!\} \equiv \boldsymbol{\mathcal{A}}(\varphi\ G)\ \mathit{in}\ v]$
using *Act-Basic-5*[*equiv-lr*] **by** *fast*
hence $[\{\!|a^P,\ G|\!\} \equiv \boldsymbol{\mathcal{A}}(\varphi\ G)\ \mathit{in}\ v]$
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 $?x = \boldsymbol{\iota}x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv (\exists\ q\ .\ q\ \&\ F = (\boldsymbol{\lambda}\ y\ .\ q)))$
have $[\Box(\exists\ p\ .\ \mathit{ContingentlyTrue}\ p)\ \mathit{in}\ v]$
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*]:
$[\Box(\varphi\ G) \to \{\!|(\boldsymbol{\iota}x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\}\ \mathit{in}\ v]$
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*]:
  $[\Box(\varphi\ G) \to \Box(\{\!|(\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\} \equiv \varphi\ G)\ in\ v]$
  proof (*rule CP*)
    assume *a*: $[\Box(\varphi\ G)\ in\ v]$
    hence $[\Box(\{\!|(\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\} \to \varphi\ G)\ in\ v]$
      using *KBasic-1*[*deduction*] **by** *simp*
    moreover {
      have $[\{\!|(\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\}\ in\ v]$
        using *a Box-desc-encode-1*[*deduction*] **by** *auto*
      hence $[\Box\{\!|(\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\}\ in\ v]$
        using *encoding*[*axiom-instance*,*deduction*] **by** *blast*
      hence $[\Box(\varphi\ G \to\ \{\!|(\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\})\ in\ v]$
        using *KBasic-1*[*deduction*] **by** *simp*
    }
    ultimately show $[\Box(\{\!|(\iota x\ .\ (\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)),\ G|\!\}$
                    $\equiv \varphi\ G)\ in\ v]$
      using **&***I KBasic-4*[*equiv-rl*] **by** *blast*
  qed

lemma *box-phi-a-1*[*PLM*]:
  assumes $[\Box(\forall\ F\ .\ \varphi\ F \to \Box(\varphi\ F))\ in\ v]$
  shows $[((\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)) \to \Box((\!|A!,x^P|\!)$
        $\&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F))\ in\ v]$
  proof (*rule CP*)
    assume *a*: $[((\!|A!,x^P|\!)\ \&\ (\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F))\ in\ v]$
    have $[\Box(\!|A!,x^P|\!)\ in\ v]$
      using *oa-facts-2*[*deduction*] *a*[*conj1*] **by** *auto*
    moreover have $[\Box(\forall\ F\ .\ \{\!|x^P,\ F|\!\} \equiv \varphi\ F)\ in\ v]$
      proof (*rule BF*[*deduction*]; *rule* ∀*I*)
        fix *F*
        have ϑ: $[\Box(\varphi\ F \to \Box(\varphi\ F))\ in\ v]$
          using *assms*[*THEN CBF*[*deduction*]] **by** (*rule* ∀*E*)
        moreover have $[\Box(\{\!|x^P,\ F|\!\} \to \Box\{\!|x^P,\ F|\!\})\ in\ v]$
          using *encoding*[*axiom-necessitation*, *axiom-instance*] **by** *simp*
        moreover have $[\Box\{\!|x^P,\ F|\!\} \equiv \Box(\varphi\ F)\ in\ v]$
          proof (*rule* ≡*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*] ∀*E* ≡*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*] ∀*E* ≡*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

TODO 5. *The proof of the following theorem seems to incorrectly reference (88) instead of (108).*

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* ∀*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 $[\forall\ y\ .\ ((\!|A!,y^P|\!)\ \&\ (\neg(\exists\ F\ .\ \{\!|y^P,\ F|\!\}))) \to y = a$
$in\ v]$
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* ∃*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 ∀*I a-prop*[*conj2*, *THEN* ∀*E*, *equiv-rl*] *id-eq-1* **by** *blast*
moreover have $[\forall\ y\ .\ ((\!|A!,y^P|\!)\ \&\ (\forall\ F\ .\ \{\!|y^P,\ F|\!\})) \to y = a\ in$
$v]$
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* ∃*I*

unfolding *Universal-def exists-unique-def* **by** *fast*
qed

lemma *null-uni-uniq-3*[*PLM*]:
[$\exists$ *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*]:
[$\exists$ *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) $\rightarrow$ $\Box$(*Null* (x^P)) *in v*]
proof (*rule CP*)
assume [*Null* (x^P) *in v*]
hence *1*: [⦇*A!*,x^P⦈ **&** ($\neg$($\exists$ *F* . ⦃x^P,*F*⦄)) *in v*]
unfolding *Null-def* **.**
have [$\Box$⦇*A!*,x^P⦈ *in v*]
using *1*[*conj1*] *oa-facts-2*[*deduction*] **by** *simp*
moreover have [$\Box$($\neg$($\exists$ *F* . ⦃x^P,*F*⦄)) *in v*]
proof −
{
assume [$\neg\Box$($\neg$($\exists$ *F* . ⦃x^P,*F*⦄)) *in v*]
hence [$\Diamond$($\exists$ *F* . ⦃x^P,*F*⦄) *in v*]
unfolding *diamond-def* **.**
hence [$\exists$ *F* . $\Diamond$⦃x^P,*F*⦄ *in v*]
using *BF$\Diamond$*[*deduction*] **by** *blast*
then obtain *P* **where** [$\Diamond$⦃x^P,*P*⦄ *in v*]
by (*rule* $\exists$ *E*)
hence [⦃x^P, *P*⦄ *in v*]
using *en-eq-3*[*equiv-lr*] **by** *simp*
hence [$\exists$ *F* . ⦃x^P, *F*⦄ *in v*]
using $\exists$ *I* **by** *blast*
}
thus *?thesis*
using *1*[*conj2*] *modus-tollens-1 CP*
useful-tautologies-1[*deduction*] **by** *metis*
qed
ultimately show [$\Box$*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) $\rightarrow$ $\Box$(*Universal* (x^P)) *in v*]
proof (*rule CP*)
assume [*Universal* (x^P) *in v*]
hence *1*: [⦇*A!*,x^P⦈ **&** ($\forall$ *F* . ⦃x^P,*F*⦄) *in v*]
unfolding *Universal-def* **.**
have [$\Box$⦇*A!*,x^P⦈ *in v*]
using *1*[*conj1*] *oa-facts-2*[*deduction*] **by** *simp*
moreover have [$\Box$($\forall$ *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* $\forall$ *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*,z^P,y^P⦈) **&** $\neg$⦃y^P, *F*⦄)) *in v*]
using *A-objects*[*axiom-instance*] **by** (*rule* $\exists$ *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* $\forall$ *E*, *THEN oth-class-taut-5-d*[*equiv-lr*],
equiv-lr]
cqt-further-4[*equiv-lr*] $\forall$ *E* **by** *blast*
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 *cut-tac* **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* ∀ *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* ∃ *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* ∀ *E*, *THEN* *oth-class-taut-5-d*[*equiv-lr*], *equiv-lr*]
cqt-further-4[*equiv-lr*] ∀ *E* **by** *blast*
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 *cut-tac* **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* ∀ *E*, *equiv-lr*] ∃ *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 [$(|A!,a^P|)$ & $(|A!,b^P|)$ & $a \neq b$
& $(\boldsymbol{\lambda}\ z\ .\ (|R,a^P,z^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,b^P,z^P|))$ *in v*]
using ϑ[*conj1*] ξ[*conj1*, *conj1*] ξ[*conj1*, *conj2*] &*I* **by** *presburger*
hence [$\exists\ y\ .\ (|A!,a^P|)$ & $(|A!,y^P|)$ & $a \neq y$
& $(\boldsymbol{\lambda}z.\ (|R,a^P,z^P|)) = (\boldsymbol{\lambda}z.\ (|R,y^P,z^P|))$ *in v*]
using ∃ *I* **by** *fast*
thus [$\exists\ x\ y\ .\ (|A!,x^P|)$ & $(|A!,y^P|)$ & $x \neq y$
& $(\boldsymbol{\lambda}z.\ (|R,x^P,z^P|)) = (\boldsymbol{\lambda}z.\ (|R,y^P,z^P|))$ *in v*]
using ∃ *I* **by** *fast*
qed

lemma *aclassical-3*[*PLM*]:
[$\forall\ F\ .\ \exists\ x\ y\ .\ (|A!,x^P|)$ & $(|A!,y^P|)$ & $(x \neq y)$
& $((\boldsymbol{\lambda}^0\ (|F,x^P|)) = (\boldsymbol{\lambda}^0\ (|F,y^P|)))$ *in v*]
proof (*rule* ∀ *I*)
fix *R*
obtain *a* **where** ϑ:

$[(|A!,a^P|)$ **&** $(\forall\ F\ .\ \{|a^P,\ F|\} \equiv (\exists\ y\ .\ (|A!,y^P|)$
& $F = (\boldsymbol{\lambda}\ z\ .\ (|R,y^P|))$ **&** $\neg\{|y^P,\ F|\}))\ \mathit{in}\ v]$
using *A-objects*[*axiom-instance*] **by** (*rule* $\exists$ *E*)
{
assume $[\neg\{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))|\}\ \mathit{in}\ v]$
hence $[\neg((|A!,a^P|)$ **&** $(\boldsymbol{\lambda}\ z\ .\ (|R,a^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))$
& $\neg\{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))|\})\ \mathit{in}\ v]$
using ϑ[*conj2*, *THEN* $\forall$ *E*, *THEN oth-class-taut-5-d*[*equiv-lr*],
equiv-lr]
cqt-further-4[*equiv-lr*] $\forall$ *E* **by** *blast*
hence $[(|A!,a^P|)$ **&** $(\boldsymbol{\lambda}\ z\ .\ (|R,a^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))$
$\rightarrow \{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))|\}\ \mathit{in}\ v]$
apply *cut-tac* **by** *PLM-solver*
hence $[\{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))|\}\ \mathit{in}\ v]$
using ϑ[*conj1*] *id-eq-1* **&***I* *vdash-properties-10* **by** *fast*
}
hence *1*: $[\{|a^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))|\}\ \mathit{in}\ v]$
using *reductio-aa-1 CP if-p-then-p* **by** *blast*
then obtain b **where** ξ:
$[(|A!,b^P|)$ **&** $(\boldsymbol{\lambda}\ z\ .\ (|R,a^P|)) = (\boldsymbol{\lambda}\ z\ .\ (|R,b^P|))$
& $\neg\{|b^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))|\}\ \mathit{in}\ v]$
using ϑ[*conj2*, *THEN* $\forall$ *E*, *equiv-lr*] $\exists$ *E* **by** *blast*
have $[a \neq b\ \mathit{in}\ v]$
proof –
{
assume $[a = b\ \mathit{in}\ v]$
hence $[\{|b^P,\ (\boldsymbol{\lambda}\ z\ .\ (|R,a^P|))|\}\ \mathit{in}\ v]$
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 $[(|R,a^P|) = (|R,b^P|)\ \mathit{in}\ v]$
unfolding *identity*$_\circ$*-def*
using ξ[*conj1*, *conj2*] **by** *auto*
hence $[(\boldsymbol{\lambda}^0\ (|R,a^P|)) = (\boldsymbol{\lambda}^0\ (|R,b^P|))\ \mathit{in}\ v]$
using *lambda-p-q-p-eq-q*[*equiv-rl*] **by** *simp*
}
ultimately have $[(|A!,a^P|)$ **&** $(|A!,b^P|)$ **&** $a \neq b$
& $((\boldsymbol{\lambda}^0\ (|R,a^P|)) = (\boldsymbol{\lambda}^0\ (|R,b^P|)))\ \mathit{in}\ v]$
using ϑ[*conj1*] ξ[*conj1*, *conj1*] ξ[*conj1*, *conj2*] **&***I*
by *presburger*
hence $[\exists\ y\ .\ (|A!,a^P|)$ **&** $(|A!,y^P|)$ **&** $a \neq y$
& $(\boldsymbol{\lambda}^0\ (|R,a^P|)) = (\boldsymbol{\lambda}^0\ (|R,y^P|))\ \mathit{in}\ v]$
using $\exists$ *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*]:

$[\exists\ x\ y\ .\ (\!|A!,x^P|\!)\ \&\ (\!|A!,y^P|\!)\ \&\ x \neq y\ \&\ (\forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!))$ *in v*]

proof −

let $?R_1 = \boldsymbol{\lambda}^2\ (\lambda\ x\ y\ .\ \forall\ F\ .\ (\!|F,x^P|\!) \equiv (\!|F,y^P|\!))$

have $[\exists\ x\ y\ .\ (\!|A!,x^P|\!)\ \&\ (\!|A!,y^P|\!)\ \&\ x \neq y$
$\&\ (\boldsymbol{\lambda} z.\ (\!|?R_1,z^P,x^P|\!)) = (\boldsymbol{\lambda} z.\ (\!|?R_1,z^P,y^P|\!))$ *in v*]

using *aclassical-1* **by** (*rule* $\forall E$)

then obtain a **where**

$[\exists\ y\ .\ (\!|A!,a^P|\!)\ \&\ (\!|A!,y^P|\!)\ \&\ a \neq y$
$\&\ (\boldsymbol{\lambda} z.\ (\!|?R_1,z^P,a^P|\!)) = (\boldsymbol{\lambda} z.\ (\!|?R_1,z^P,y^P|\!))$ *in v*]

by (*rule* $\exists E$)

then obtain b **where** *ab-prop*:

$[(\!|A!,a^P|\!)\ \&\ (\!|A!,b^P|\!)\ \&\ a \neq b$
$\&\ (\boldsymbol{\lambda} z.\ (\!|?R_1,z^P,a^P|\!)) = (\boldsymbol{\lambda} z.\ (\!|?R_1,z^P,b^P|\!))$ *in v*]

by (*rule* $\exists E$)

have $[(\!|?R_1,\ a^P,\ a^P|\!)$ *in v*]

apply (*rule beta-C-meta-2*[*equiv-rl*])

apply (*rule IsPropositional-intros*)

using *oth-class-taut-4-a*[*THEN* $\forall I$] **by** *fast*

hence $[(\!|\boldsymbol{\lambda}\ z\ .\ (\!|?R_1,\ z^P,\ a^P|\!),\ a^P|\!)$ *in v*]

apply *cut-tac* **apply** (*rule beta-C-meta-1*[*equiv-rl*])

apply (*rule IsPropositional-intros*)

by *auto*

hence $[(\!|\boldsymbol{\lambda}\ z\ .\ (\!|?R_1,\ z^P,\ b^P|\!),\ a^P|\!)$ *in v*]

using *ab-prop*[*conj2*] *l-identity*[*axiom-instance*, *deduction*, *deduction*]

by *fast*

hence $[(\!|?R_1,\ a^P,\ b^P|\!)$ *in v*]

using *beta-C-meta-1*[*equiv-lr*] *IsPropositional-intros* **by** *fast*

hence $[\forall F.\ (\!|F,a^P|\!) \equiv (\!|F,b^P|\!)$ *in v*]

using *beta-C-meta-2*[*equiv-lr*] *IsPropositional-intros* **by** *fast*

hence $[(\!|A!,a^P|\!)\ \&\ (\!|A!,b^P|\!)\ \&\ a \neq b\ \&\ (\forall F.\ (\!|F,a^P|\!) \equiv (\!|F,b^P|\!))$ *in v*]

using *ab-prop*[*conj1*] **&***I* **by** *presburger*

hence $[\exists\ y\ .\ (\!|A!,a^P|\!)\ \&\ (\!|A!,y^P|\!)\ \&\ a \neq y\ \&\ (\forall F.\ (\!|F,a^P|\!) \equiv (\!|F,y^P|\!))$ *in v*]

using $\exists I$ **by** *fast*

thus *?thesis* **using** $\exists I$ **by** *fast*

qed

9.13 Propositional Properties

lemma *prop-prop2-1*:

$[\forall\ p\ .\ \exists\ F\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)$ *in v*]

proof (*rule* $\forall I$)

fix p

have $[(\boldsymbol{\lambda}\ x\ .\ p) = (\boldsymbol{\lambda}\ x\ .\ p)$ *in v*]

using *id-eq-prop-prop-1* **by** *auto*

thus $[\exists\ F\ .\ F = (\boldsymbol{\lambda}\ x\ .\ p)$ *in v*]

by *PLM-solver*

qed

lemma *prop-prop2-2*:

$[F = (\boldsymbol{\lambda}\ x\ .\ p) \rightarrow \Box(\forall\ x\ .\ (\!|F,x^P|\!) \equiv p)$ *in v*]

```
proof (rule CP)
  assume 1: [F = (λ x . p) in v]
  {
    fix v
    {
      fix x
      have [⦇(λ x . p), x^P⦈ ≡ p in v]
        apply (rule beta-C-meta-1)
        by (rule IsPropositional-intros)+
    }
    hence [∀ x . ⦇(λ x . p), x^P⦈ ≡ p in v]
      by (rule ∀I)
  }
  hence [□(∀ x . ⦇(λ x . p), x^P⦈ ≡ p) in v]
    by (rule RN)
  thus [□(∀x. ⦇F,x^P⦈ ≡ 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 → □(Propositional F) in v]
  proof (rule CP)
    assume [Propositional F in v]
    hence [∃ p . F = (λ x . p) in v]
      unfolding Propositional-def .
    then obtain q where [F = (λ x . q) in v]
      by (rule ∃E)
    hence [□(F = (λ x . q)) in v]
      using id-nec[equiv-lr] by auto
    hence [∃ p . □(F = (λ x . p)) in v]
      using ∃I by fast
    thus [□(Propositional F) in v]
      unfolding Propositional-def
      using sign-S5-thm-1[deduction] by fast
  qed


lemma prop-indis:
  [Indiscriminate F → (¬(∃ x y . ⦇F,x^P⦈ & (¬⦇F,y^P⦈))) in v]
  proof (rule CP)
    assume [Indiscriminate F in v]
    hence 1: [□((∃x. ⦇F,x^P⦈) → (∀x. ⦇F,x^P⦈)) in v]
      unfolding Indiscriminate-def .
    {
      assume [∃ x y . ⦇F,x^P⦈ & ¬⦇F,y^P⦈ in v]
      then obtain x where [∃ y . ⦇F,x^P⦈ & ¬⦇F,y^P⦈ in v]
        by (rule ∃E)
      then obtain y where 2: [⦇F,x^P⦈ & ¬⦇F,y^P⦈ in v]
        by (rule ∃E)
      hence [∃ x . ⦇F, x^P⦈ in v]
        using &E(1) ∃I by fast
      hence [∀ x . ⦇F,x^P⦈ in v]
        using 1[THEN qml-2[axiom-instance, deduction], deduction] by
```

fast

hence [⦇F,y^P⦈ *in v*]
  using *cqt-orig-1*[*deduction*] **by** *fast*
hence [⦇F,y^P⦈ **&** (¬⦇F,y^P⦈) *in v*]
  using *2* **&***I* **&***E* **by** *fast*
hence [¬(∃ *x y* . ⦇F,x^P⦈ **&** ¬⦇F,y^P⦈) *in v*]
  using *pl-1*[*axiom-instance*, *deduction*, *THEN modus-tollens-1*]
    oth-class-taut-1-a **by** *blast*
}
thus [¬(∃ *x y* . ⦇F,x^P⦈ **&** ¬⦇F,y^P⦈) *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*]
      using *beta-C-meta-1*[*equiv-lr*] *IsPropositional-intros* **by** *fast*
    {
      fix *x*
      have [⦇**λ** *y* . *q*, x^P⦈ *in w*]
        using *q beta-C-meta-1*[*equiv-rl*] *IsPropositional-intros* **by** *fast*
      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 cut-tac 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-subst1-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-subst1-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 cut-tac
      apply (PLM-subst1-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 *cut-tac*
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,**λ***x*. *p*⦄) *in v*]
unfolding *identity-defs* **.**
hence [□(∀ *x*. ⦃x^P,*F*⦄ ≡ ⦃x^P,**λ***x*. *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* = (**λ***x*. *p*))
□¬(∃ *p*. *F* = (**λ***x*. *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*:
[◇(∀ *F* . ⦃x^P, *F*⦄ → (∃ *p* . *F* = (**λ** *x* . *p*)))
→ (∀ *F* . ⦃x^P, *F*⦄ → (∃ *p* . *F* = (**λ** *x* . *p*))) *in v*]
proof (*rule CP*)
assume [◇(∀ *F*. ⦃x^P,*F*⦄ → (∃ *p*. *F* = (**λ***x*. *p*))) *in v*]
hence *1*: [(∀ *F*. ◇(⦃x^P,*F*⦄ → (∃ *p*. *F* = (**λ***x*. *p*)))) *in v*]
using *Buridan*◇[*deduction*] **by** *auto*
{
fix *Q*
assume [⦃x^P,*Q*⦄ *in v*]
hence [□⦃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** *auto*
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 *cut-tac* **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 Sanity Tests

10.1 Consistency

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

10.2 Intensionality

context
begin
interpretation *MetaSolver*.

lemma $[(\boldsymbol{\lambda} y.\ (q \lor \neg q)) = (\boldsymbol{\lambda} y.\ (p \lor \neg p))\ in\ v]$
unfolding *identity*-Π_1-*def*
apply (*rule* $Eq_1 I$) **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 $(i \Rightarrow bool) \times i = 4$,
card $(i \Rightarrow bool) \times (i \Rightarrow bool) \times i = 4$,
card υ *= 2, verbose, show-all, debug*]
oops — Countermodel by Nitpick
lemma $[(\boldsymbol{\lambda} y.\ (p \lor q)) = (\boldsymbol{\lambda} y.\ (q \lor p))\ in\ v]$
unfolding *identity*-Π_1-*def*
apply (*rule* $Eq_1 I$) **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* $(i \Rightarrow bool) \times i = 4$,
card $(i \Rightarrow bool) \times (i \Rightarrow bool) \times i = 4$,
card υ *= 2, verbose, show-all, debug*]

oops — Countermodel by Nitpick
end

10.3 Concreteness coindices with Object Domains

context
begin
private lemma *OrdCheck*:
$[(\!|\boldsymbol{\lambda}\ x\ .\ \neg\Box(\neg(\!|E!,\ x^P|\!))),\ x|\!)\ in\ v] \longleftrightarrow$
$(denotes\ x) \wedge (case\ (denotation\ x)\ of\ \omega\nu\ y \Rightarrow True\ |\ \text{-} \Rightarrow False)$
using *OrdinaryObjectsPossiblyConcreteAxiom*
by (*simp add*: *meta-defs meta-aux split*: *ν.split υ.split*)
private lemma *AbsCheck*:
$[(\!|\boldsymbol{\lambda}\ x\ .\ \Box(\neg(\!|E!,\ x^P|\!)),\ x|\!)\ in\ v] \longleftrightarrow$
$(denotes\ x) \wedge (case\ (denotation\ x)\ of\ \alpha\nu\ y \Rightarrow True\ |\ \text{-} \Rightarrow False)$
using *OrdinaryObjectsPossiblyConcreteAxiom*
by (*simp add*: *meta-defs meta-aux split*: *ν.split υ.split*)
end

10.4 Justification for Meta-Logical Axioms

context
begin

Remark 24. *OrdinaryObjectsPossiblyConcreteAxiom is equivalent to "all ordinary objects are possibly concrete".*

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

Remark 25. *OrdinaryObjectsPossiblyConcreteAxiom is equivalent to "all abstract objects are necessarily not concrete".*

private lemma *AbsAxiomCheck*:
OrdinaryObjectsPossiblyConcrete $\longleftrightarrow$
$(\forall\ x.\ ([(\!|\boldsymbol{\lambda}\ x\ .\ \Box(\neg(\!|E!,\ x^P|\!)),\ x^P|\!)\ in\ v]$
$\longleftrightarrow (case\ x\ of\ \alpha\nu\ y \Rightarrow True\ |\ \text{-} \Rightarrow False)))$
by (*auto simp*: *meta-defs meta-aux split*: *ν.split υ.split*)

Remark 26. *PossiblyContingentObjectExistsAxiom is equivalent to the corresponding statement in the embedded logic.*

private lemma *PossiblyContingentObjectExistsCheck*:
$[\neg(\Box(\forall\ x.\ (\!|E!,x^P|\!) \rightarrow \Box(\!|E!,x^P|\!)))\ in\ v]$
apply (*simp add*: *meta-defs forall-ν-def meta-aux split*: *ν.split υ.split*)
using *PossiblyContingentObjectExistsAxiom*
by (*metis ν.simps(5) νυ-def υ.simps(1) no-σω*)
private lemma *PossiblyContingentObjectExists*
apply (*auto simp*: *meta-defs*)
using *PossiblyContingentObjectExistsCheck*
apply (*auto simp*: *meta-defs forall-ν-def meta-aux split*: *ν.split υ.split*)

by (*metis* υ*.exhaust* υ*.simps*(*5*) υ*.simps*(*6*))

Remark 27. *PossiblyNoContingentObjectExistsAxiom is equivalent to the corresponding statement in the embedded logic.*

private lemma *PossiblyNoContingentObjectExistsCheck*:
$[\neg(\Box(\neg(\forall\ x.\ (|E!,x^P|) \rightarrow \Box(|E!,x^P|)))) \ in\ v]$
apply (*simp add*: *meta-defs forall-*ν*-def meta-aux split*: ν*.split* υ*.split*)
using *PossiblyNoContingentObjectExistsAxiom* **by** *blast*
private lemma *PossiblyNoContingentObjectExists*
using *PossiblyNoContingentObjectExistsCheck*
apply (*auto simp*: *meta-defs forall-*ν*-def meta-aux split*: ν*.split* υ*.split*)
by (*metis* υ*.simps*(*5*) $\nu\upsilon$-$\upsilon\nu$*-id*)
end

10.5 Relations in the Meta-Logic

context
begin

Remark 28. *Material equality in the embedded logic corresponds to equality in the actual state in the meta-logic.*

private lemma *mat-eq-is-eq-dj*:
$[\forall\ x\ .\ \Box((|F,x^P|) \equiv (|G,x^P|))\ in\ v] \longleftrightarrow$
$((\lambda\ x\ .\ (eval\Pi_1\ F)\ x\ dj) = (\lambda\ x\ .\ (eval\Pi_1\ G)\ x\ dj))$
proof
interpret *MetaSolver* **.**
interpret *Semantics* **.**
assume *1*: $[\forall x.\ \Box((|F,x^P|) \equiv (|G,x^P|))\ in\ v]$
{
fix v
fix y
obtain x **where** *y-def*: $y = \nu\upsilon\ x$ **by** (*metis* $\nu\upsilon$-$\upsilon\nu$*-id*)
have $(\exists\, r\ o_1.\ Some\ r = d_1\ F \wedge Some\ o_1 = d_\kappa\ (x^P) \wedge o_1 \in ex1\ r$
$v) =$
$(\exists\, r\ o_1.\ Some\ r = d_1\ G \wedge Some\ o_1 = d_\kappa\ (x^P) \wedge o_1 \in ex1\ r\ v)$
using *1* **apply** *cut-tac* **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
interpret *MetaSolver* **.**
interpret *Semantics* **.**

```
    assume 1: (λx. evalΠ₁ F x dj) = (λx. evalΠ₁ G x dj)
    {
      fix y v
      obtain x where x-def: x = νυ y
        by simp
      hence evalΠ₁ F x dj = evalΠ₁ G x dj
        using 1 by metis
      moreover obtain r where r-def: Some r = d₁ F
        unfolding d₁-def by auto
      moreover obtain s where s-def: Some s = d₁ G
        unfolding d₁-def by auto
      ultimately have (y ∈ ex1 r v) = (y ∈ ex1 s v)
        by (simp add: d₁.rep-eq ex1-def νυ-υν-id x-def)
      hence [(|F,y^P|) ≡ (|G,y^P|) in v]
        apply cut-tac apply meta-solver
        using r-def s-def by (metis Semantics.d_κ-proper option.inject)
    }
    thus [∀ x. □((|F,x^P|) ≡ (|G,x^P|)) in v]
      using T6 T8 by fast
  qed
```

Remark 29. *Material equivalent relations are equal in the embedded logic if and only if they also coincide in all other states.*

```
  private lemma mat-eq-is-eq-if-eq-forall-j:
    assumes [∀ x . □((|F,x^P|) ≡ (|G,x^P|)) in v]
    shows [F = G in v] ⟷
          (∀ s . s ≠ dj ⟶ (∀ x . (evalΠ₁ F) x s = (evalΠ₁ G) x s))
    proof
      interpret MetaSolver .
      assume [F = G in v]
      hence F = G
        apply cut-tac unfolding identity-Π₁-def by meta-solver
      thus ∀s. s ≠ dj ⟶ (∀x. evalΠ₁ F x s = evalΠ₁ G x s)
        by auto
    next
      interpret MetaSolver .
      assume ∀s. s ≠ dj ⟶ (∀x. evalΠ₁ F x s = evalΠ₁ G x s)
     moreover have ((λ x . (evalΠ₁ F) x dj) = (λ x . (evalΠ₁ G) x dj))
        using assms mat-eq-is-eq-dj by auto
      ultimately have ∀s x. evalΠ₁ F x s = evalΠ₁ G x s
        by metis
      hence evalΠ₁ F = evalΠ₁ G
        by blast
      hence F = G
        by (metis evalΠ₁-inverse)
      thus [F = G in v]
        unfolding identity-Π₁-def using Eq₁I by auto
    qed
```

Remark 30. *Under the assumption that all properties behave in all states like in the actual state the defined equality degenerates to material equality.*

```
  lemma assumes ∀ F x s . (evalΠ₁ F) x s = (evalΠ₁ 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)

end