

Embedding of the Theory of Abstract Objects in Isabelle/HOL

Daniel Kirchner

March 6, 2017

Abstract

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

Contents

1	Embedding	3
1.1	Primitives	3
1.2	Mapping from abstract objects to special Urelements	4
1.3	Conversion between objects and Urelements	4
1.4	Exemplification of n-place relations.	4
1.5	Encoding	5
1.6	Connectives and Quantifiers	5
1.7	Definite Description	5
1.8	Lambda Expressions	6
1.9	Validity	6
1.10	Concreteness	6
1.11	Automation	7
1.12	Auxiliary Lemmata	7
2	Basic Definitions	8
2.1	Derived Connectives	8
2.2	Abstract and Ordinary Objects	8
2.3	Identity Definitions	8
3	Semantics	9
3.1	Propositional Formulas	9
3.2	Semantics	13
3.3	Validity Syntax	17
4	MetaSolver	17
4.1	Rules for Implication	17
4.2	Rules for Negation	17
4.3	Rules for Conjunction	18
4.4	Rules for Equivalence	18
4.5	Rules for Disjunction	18

4.6	Rules for Necessity	18
4.7	Rules for Possibility	18
4.8	Rules for Quantification	18
4.9	Rules for Actuality	19
4.10	Rules for Encoding	19
4.11	Rules for Exemplification	20
4.11.1	Zero-place Relations	20
4.11.2	One-Place Relations	20
4.11.3	Two-Place Relations	20
4.11.4	Three-Place Relations	20
4.12	Rules for Being Ordinary	21
4.13	Rules for Being Abstract	21
4.14	Rules for Definite Descriptions	22
4.15	Rules for Identity	22
4.15.1	Ordinary Objects	22
4.15.2	Individuals	23
4.15.3	One-Place Relations	25
4.15.4	Two-Place Relations	25
4.15.5	Three-Place Relations	25
4.15.6	Propositions	26
5	General Quantification	26
5.1	Type Class	26
5.2	Instantiations	27
5.3	MetaSolver Rules	28
5.3.1	Rules for General All Quantification.	28
5.3.2	Rules for Existence	29
6	General Identity	29
6.1	Type Classes	29
6.2	Instantiations	29
6.3	New Identity Definitions	31
7	The Axioms of Principia Metaphysica	32
7.1	Closures	32
7.2	Axioms for Negations and Conditionals	33
7.3	Axioms of Identity	33
7.4	Axioms of Quantification	33
7.5	Axioms of Actuality	35
7.6	Axioms of Necessity	35
7.7	Axioms of Necessity and Actuality	36
7.8	Axioms of Descriptions	36
7.9	Axioms for Complex Relation Terms	37
7.10	Axioms of Encoding	38
8	Definitions	39
8.1	Property Negations	39
8.2	Noncontingent and Contingent Relations	39
8.3	Null and Universal Objects	40
8.4	Propositional Properties	41
8.5	Indiscriminate Properties	41
8.6	Miscellaneous	41

9	The Deductive System PLM	41
9.1	Automatic Solver	41
9.2	Modus Ponens	41
9.3	Axioms	42
9.4	(Modally Strict) Proofs and Derivations	42
9.5	GEN and RN	42
9.6	Negations and Conditionals	42
9.7	Identity	50
9.8	Quantification	57
9.9	Actuality and Descriptions	59
9.10	Necessity	72
9.11	The Theory of Relations	92
9.12	The Theory of Objects	121
9.13	Propositional Properties	140
10	Possible Worlds	145
10.1	Definitions	145
10.2	Auxiliary Lemmata	146
10.3	For every syntactic Possible World there is a semantic Possible World	148
10.4	For every semantic Possible World there is a syntactic Possible World	150
11	Sanity Tests	151
11.1	Consistency	151
11.2	Intensionality	151
11.3	Concreteness coindices with Object Domains	152
11.4	Justification for Meta-Logical Axioms	152
11.5	Relations in the Meta-Logic	153

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  $\omega$  — ordinary objects
typedecl  $\sigma$  — special Urelements
datatype  $v = \omega v \ \omega \mid \sigma v \ \sigma$  — Urelements

type-synonym  $\Pi_0 = o$  — zero place relations
typedef  $\Pi_1 = UNIV :: (v \Rightarrow j \Rightarrow i \Rightarrow bool)$  set
  morphisms evalPi1 makePi1 .. — one place relations
typedef  $\Pi_2 = UNIV :: (v \Rightarrow v \Rightarrow j \Rightarrow i \Rightarrow bool)$  set
  morphisms evalPi2 makePi2 .. — two place relations
typedef  $\Pi_3 = UNIV :: (v \Rightarrow v \Rightarrow v \Rightarrow j \Rightarrow i \Rightarrow bool)$  set
  morphisms evalPi3 makePi3 .. — 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 $\kappa = \text{UNIV}::(\text{bool} \times \nu)$ 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 explicitly 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 :: $\kappa \Rightarrow \text{bool}$ 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 $\nu \nu :: \nu \Rightarrow \nu$ **where** $\nu \nu \equiv \text{case-}\nu \ \omega \nu \ (\sigma \nu \circ \alpha \sigma)$

definition $v \nu :: v \Rightarrow \nu$ **where** $v \nu \equiv \text{case-}v \ \omega \nu \ (\alpha \nu \circ (\text{inv } \alpha \sigma))$

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 :: $\Pi_0 \Rightarrow o$ ($\langle _, _ \rangle$) is id .

lift-definition exe1 :: $\Pi_1 \Rightarrow \kappa \Rightarrow o$ ($\langle _, _ \rangle$) is

$\lambda F \ x \ w \ s . (\text{denotes } x) \wedge F (\nu \nu (\text{denotation } x)) \ w \ s .$

lift-definition exe2 :: $\Pi_2 \Rightarrow \kappa \Rightarrow \kappa \Rightarrow o$ ($\langle _, _, _ \rangle$) is

$\lambda F \ x \ y \ w \ s . (\text{denotes } x) \wedge (\text{denotes } y) \wedge$
 $F (\nu \nu (\text{denotation } x)) (\nu \nu (\text{denotation } y)) \ w \ s .$

lift-definition exe3 :: $\Pi_3 \Rightarrow \kappa \Rightarrow \kappa \Rightarrow \kappa \Rightarrow o$ ($\langle _, _, _, _ \rangle$) is

$\lambda F \ x \ y \ z \ w \ s . (\text{denotes } x) \wedge (\text{denotes } y) \wedge (\text{denotes } z) \wedge$

$F (\nu \nu (\text{denotation } x)) (\nu \nu (\text{denotation } y)) (\nu \nu (\text{denotation } z)) \ w \ s .$

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 $enc :: \kappa \Rightarrow \Pi_1 \Rightarrow o$ ($\llbracket -, - \rrbracket$) is
 $\lambda x F w s . (denotes\ x) \wedge case\text{-}\nu\ (\lambda \omega . False)\ (\lambda \alpha . F \in \alpha)\ (denotation\ x) .$

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 $not :: o \Rightarrow o$ (\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 $impl :: o \Rightarrow o \Rightarrow o$ (**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 $forall_\nu :: (\nu \Rightarrow o) \Rightarrow o$ (**binder** \forall_ν [8] 9) is
 $\lambda \varphi s w . \forall x :: \nu . (\varphi\ x)\ s\ w .$

lift-definition $forall_0 :: (\Pi_0 \Rightarrow o) \Rightarrow o$ (**binder** \forall_0 [8] 9) is
 $\lambda \varphi s w . \forall x :: \Pi_0 . (\varphi\ x)\ s\ w .$

lift-definition $forall_1 :: (\Pi_1 \Rightarrow o) \Rightarrow o$ (**binder** \forall_1 [8] 9) is
 $\lambda \varphi s w . \forall x :: \Pi_1 . (\varphi\ x)\ s\ w .$

lift-definition $forall_2 :: (\Pi_2 \Rightarrow o) \Rightarrow o$ (**binder** \forall_2 [8] 9) is
 $\lambda \varphi s w . \forall x :: \Pi_2 . (\varphi\ x)\ s\ w .$

lift-definition $forall_3 :: (\Pi_3 \Rightarrow o) \Rightarrow o$ (**binder** \forall_3 [8] 9) is
 $\lambda \varphi s w . \forall x :: \Pi_3 . (\varphi\ x)\ s\ w .$

lift-definition $forall_o :: (o \Rightarrow o) \Rightarrow o$ (**binder** \forall_o [8] 9) is
 $\lambda \varphi s w . \forall x :: o . (\varphi\ x)\ s\ w .$

lift-definition $box :: o \Rightarrow o$ (\Box - [62] 63) is
 $\lambda p s w . \forall v . p\ s\ v .$

lift-definition $actual :: o \Rightarrow o$ (\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 $that :: (\nu \Rightarrow o) \Rightarrow \kappa$ (**binder** ι [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 $\nu\nu$ 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. $\lambda x . \llbracket x^P, F \rrbracket$ 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* :: $\text{o} \Rightarrow \Pi_0 (\lambda^0)$ is *id* .
lift-definition *lambdabinder1* :: $(\nu \Rightarrow \text{o}) \Rightarrow \Pi_1 (\text{binder } \lambda [8] 9)$ is
 $\lambda \varphi u . \varphi (\nu\nu u)$.
lift-definition *lambdabinder2* :: $(\nu \Rightarrow \nu \Rightarrow \text{o}) \Rightarrow \Pi_2 (\lambda^2)$ is
 $\lambda \varphi u v . \varphi (\nu\nu u) (\nu\nu v)$.
lift-definition *lambdabinder3* :: $(\nu \Rightarrow \nu \Rightarrow \nu \Rightarrow \text{o}) \Rightarrow \Pi_3 (\lambda^3)$ is
 $\lambda \varphi u v w . \varphi (\nu\nu u) (\nu\nu v) (\nu\nu w)$.

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* :: $i \Rightarrow \text{o} \Rightarrow \text{bool}$ (**infixl** \models 5) is
 $\lambda v \varphi . \varphi \text{ dj } v$.

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* :: $\omega \Rightarrow i \Rightarrow \text{bool}$

abbreviation *OrdinaryObjectsPossiblyConcrete* **where**

OrdinaryObjectsPossiblyConcrete $\equiv \forall x . \exists v . \text{ConcreteInWorld } x v$

abbreviation *PossiblyContingentObjectExists* **where**

PossiblyContingentObjectExists $\equiv \exists x v . \text{ConcreteInWorld } x v$
 $\wedge (\exists w . \neg \text{ConcreteInWorld } x w)$

abbreviation *PossiblyNoContingentObjectExists* **where**

$$\text{PossiblyNoContingentObjectExists} \equiv \exists w . \forall x . \text{ConcreteInWorld } x w \\ \longrightarrow (\forall v . \text{ConcreteInWorld } x v)$$

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*

$\lambda u s w . \text{case } u \text{ of } \omega v x \Rightarrow \text{ConcreteInWorld } x w \mid - \Rightarrow \text{False} .$

1.11 Automation

named-theorems *meta-defs*

declare *not-def*[*meta-defs*] *impl-def*[*meta-defs*] *forall_ν-def*[*meta-defs*]
forall₀-def[*meta-defs*] *forall₁-def*[*meta-defs*]
forall₂-def[*meta-defs*] *forall₃-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 *makeκ-inverse*[*meta-aux*] *evalκ-inverse*[*meta-aux*]
makeo-inverse[*meta-aux*] *evalo-inverse*[*meta-aux*]
makeΠ₁-inverse[*meta-aux*] *evalΠ₁-inverse*[*meta-aux*]
makeΠ₂-inverse[*meta-aux*] *evalΠ₂-inverse*[*meta-aux*]
makeΠ₃-inverse[*meta-aux*] *evalΠ₃-inverse*[*meta-aux*]
lemma *νν-ων-is-ων*[*meta-aux*]: $\nu\nu (\omega\nu x) = \omega\nu x$ **by** (*simp add: νν-def*)
lemma *νν-ων-is-ων*[*meta-aux*]: $\nu\nu (\omega\nu x) = \omega\nu x$ **by** (*simp add: νν-def*)
lemma *denotation-proper*[*meta-aux*]: *denotation* (x^P) = x
by (*simp add: meta-aux νκ-def denotation-def*)
lemma *proper-denotes*[*meta-aux*]: *denotes* (x^P)
by (*simp add: meta-aux νκ-def denotes-def*)
lemma *proper-denotation*[*meta-aux*]: *denotation* (x^P) = x
by (*simp add: meta-aux νκ-def denotation-def*)
lemma *νν-νν-id*[*meta-aux*]: $\nu\nu (\nu\nu (x)) = x$

by (simp add: $\nu\nu$ -def $\nu\nu$ -def $\alpha\sigma$ -surj surj-f-inv-f split: v .split)
lemma no- $\alpha\omega$ [meta-aux]: $\neg(\nu\nu (\alpha\nu x) = \omega\nu y)$ **by** (simp add: $\nu\nu$ -def)
lemma no- $\sigma\omega$ [meta-aux]: $\neg(\sigma\nu x = \omega\nu y)$ **by** blast
lemma $\nu\nu$ -surj[meta-aux]: surj $\nu\nu$ **using** $\nu\nu$ - $\nu\nu$ -id surjI **by** blast
lemma $\nu\nu\kappa$ -aux1[meta-aux]:
 fst (eval κ ($\nu\nu$ ($\nu\nu$ (snd (eval κ x)))^P))
 apply transfer
 by simp
lemma $\nu\nu\kappa$ -aux2[meta-aux]:
 ($\nu\nu$ (snd (eval κ ($\nu\nu$ ($\nu\nu$ (snd (eval κ x)))^P)))) = ($\nu\nu$ (snd (eval κ x)))
 apply transfer
 using $\nu\nu$ - $\nu\nu$ -id **by** auto

2 Basic Definitions

2.1 Derived Connectives

definition diamond:: $\circ \Rightarrow \circ$ (\Diamond - [62] 63) **where**

diamond $\equiv \lambda \varphi . \neg \Box \neg \varphi$

definition conj:: $\circ \Rightarrow \circ \Rightarrow \circ$ (infixl & 53) **where**

conj $\equiv \lambda x y . \neg(x \rightarrow \neg y)$

definition disj:: $\circ \Rightarrow \circ \Rightarrow \circ$ (infixl \vee 52) **where**

disj $\equiv \lambda x y . \neg x \rightarrow y$

definition equiv:: $\circ \Rightarrow \circ \Rightarrow \circ$ (infixl \equiv 51) **where**

equiv $\equiv \lambda x y . (x \rightarrow y) \ \& \ (y \rightarrow x)$

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 $\equiv \lambda x . \Diamond \langle E!, x^P \rangle$

definition Abstract :: Π_1 ($A!$) **where** Abstract $\equiv \lambda x . \neg \Diamond \langle E!, x^P \rangle$

2.3 Identity Definitions

definition basic-identity_E:: Π_2 **where**

basic-identity_E $\equiv \lambda^2 (\lambda x y . \langle O!, x^P \rangle \ \& \ \langle O!, y^P \rangle$
 $\ \& \ \Box (\forall_1 F . \langle F, x^P \rangle \equiv \langle F, y^P \rangle))$

definition basic-identity_E-infix:: $\kappa \Rightarrow \kappa \Rightarrow \circ$ (infixl =_E 63) **where**

$x =_E y \equiv \langle \text{basic-identity}_E, x, y \rangle$

definition basic-identity _{κ} (infixl = _{κ} 63) **where**

basic-identity _{κ} $\equiv \lambda x y . (x =_E y) \vee \langle A!, x \rangle \ \& \ \langle A!, y \rangle$
 $\ \& \ \Box (\forall_1 F . \langle x, F \rangle \equiv \langle y, F \rangle)$

definition basic-identity₁ (infixl =₁ 63) **where**

basic-identity₁ $\equiv \lambda F G . \Box (\forall_\nu x . \langle x^P, F \rangle \equiv \langle x^P, G \rangle)$

definition basic-identity₂ :: $\Pi_2 \Rightarrow \Pi_2 \Rightarrow \circ$ (infixl =₂ 63) **where**

basic-identity₂ $\equiv \lambda F G . \forall_\nu x . ((\lambda y . \langle F, x^P, y^P \rangle) =_1 (\lambda y . \langle G, x^P, y^P \rangle))$

$$\& ((\lambda y. \langle F, y^P, x^P \rangle) =_1 (\lambda y. \langle G, y^P, x^P \rangle))$$

definition *basic-identity₃* :: $\Pi_3 \Rightarrow \Pi_3 \Rightarrow \circ$ (**infixl** =₃ 63) **where**

$$\begin{aligned} \text{basic-identity}_3 &\equiv \lambda F G . \forall \nu x y . (\lambda z . \langle F, z^P, x^P, y^P \rangle) =_1 (\lambda z . \langle G, z^P, x^P, y^P \rangle) \\ &\& (\lambda z . \langle F, x^P, z^P, y^P \rangle) =_1 (\lambda z . \langle G, x^P, z^P, y^P \rangle) \\ &\& (\lambda z . \langle F, x^P, y^P, z^P \rangle) =_1 (\lambda z . \langle G, x^P, y^P, z^P \rangle) \end{aligned}$$

definition *basic-identity_o* :: $\circ \Rightarrow \circ \Rightarrow \circ$ (**infixl** =_o 63) **where**

$$\text{basic-identity}_o \equiv \lambda F G . (\lambda y . F) =_1 (\lambda y . G)$$

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* :: $(\kappa \Rightarrow \circ) \Rightarrow \text{bool}$ **where**

$$\begin{aligned} \text{IsPropositionalInX} &\equiv \lambda \Theta . \exists \chi . \Theta = (\lambda x . \chi \\ & \quad (* \text{ one place } *) (\lambda F . \langle F, x \rangle) \\ & \quad (* \text{ two place } *) (\lambda F . \langle F, x, x \rangle) (\lambda F a . \langle F, x, a \rangle) (\lambda F a . \langle F, a, x \rangle) \\ & \quad (* \text{ three place three } x *) (\lambda F . \langle F, x, x, x \rangle) \\ & \quad (* \text{ three place two } x *) (\lambda F a . \langle F, x, x, a \rangle) (\lambda F a . \langle F, x, a, x \rangle) \\ & \quad \quad (\lambda F a . \langle F, a, x, x \rangle) \\ & \quad (* \text{ three place one } x *) (\lambda F a b . \langle F, x, a, b \rangle) (\lambda F a b . \langle F, a, x, b \rangle) \\ & \quad \quad (\lambda F a b . \langle F, a, b, x \rangle)) \end{aligned}$$

lemma *IsPropositionalInX-intro*[*IsPropositional-intros*]:

$$\begin{aligned} \text{IsPropositionalInX} &(\lambda x . \chi \\ & \quad (* \text{ one place } *) (\lambda F . \langle F, x \rangle) \\ & \quad (* \text{ two place } *) (\lambda F . \langle F, x, x \rangle) (\lambda F a . \langle F, x, a \rangle) (\lambda F a . \langle F, a, x \rangle) \\ & \quad (* \text{ three place three } x *) (\lambda F . \langle F, x, x, x \rangle) \\ & \quad (* \text{ three place two } x *) (\lambda F a . \langle F, x, x, a \rangle) (\lambda F a . \langle F, x, a, x \rangle) \\ & \quad \quad (\lambda F a . \langle F, a, x, x \rangle) \\ & \quad (* \text{ three place one } x *) (\lambda F a b . \langle F, x, a, b \rangle) (\lambda F a b . \langle F, a, x, b \rangle) \\ & \quad \quad (\lambda F a b . \langle F, a, b, x \rangle)) \end{aligned}$$

unfolding *IsPropositionalInX-def* **by** *blast*

definition *IsPropositionalInXY* :: $(\kappa \Rightarrow \kappa \Rightarrow \circ) \Rightarrow \text{bool}$ **where**

$$\begin{aligned} \text{IsPropositionalInXY} &\equiv \lambda \Theta . \exists \chi . \Theta = (\lambda x y . \chi \\ & \quad (* \text{ only } x *) \\ & \quad \quad (* \text{ one place } *) (\lambda F . \langle F, x \rangle) \\ & \quad \quad (* \text{ two place } *) (\lambda F . \langle F, x, x \rangle) (\lambda F a . \langle F, x, a \rangle) (\lambda F a . \langle F, a, x \rangle) \\ & \quad \quad (* \text{ three place three } x *) (\lambda F . \langle F, x, x, x \rangle) \\ & \quad \quad (* \text{ three place two } x *) (\lambda F a . \langle F, x, x, a \rangle) (\lambda F a . \langle F, x, a, x \rangle) \\ & \quad \quad \quad (\lambda F a . \langle F, a, x, x \rangle) \\ & \quad \quad (* \text{ three place one } x *) (\lambda F a b . \langle F, x, a, b \rangle) (\lambda F a b . \langle F, a, x, b \rangle) \end{aligned}$$

$(\lambda F a b . \langle F, a, b, x \rangle)$
 (* only y *)
 (* one place *) $(\lambda F . \langle F, y \rangle)$
 (* two place *) $(\lambda F . \langle F, y, y \rangle) (\lambda F a . \langle F, y, a \rangle) (\lambda F a . \langle F, a, y \rangle)$
 (* three place three y *) $(\lambda F . \langle F, y, y, y \rangle)$
 (* three place two y *) $(\lambda F a . \langle F, y, y, a \rangle) (\lambda F a . \langle F, y, a, y \rangle)$
 $(\lambda F a . \langle F, a, y, y \rangle)$
 (* three place one y *) $(\lambda F a b . \langle F, y, a, b \rangle) (\lambda F a b . \langle F, a, y, b \rangle)$
 $(\lambda F a b . \langle F, a, b, y \rangle)$
 (* x and y *)
 (* two place *) $(\lambda F . \langle F, x, y \rangle) (\lambda F . \langle F, y, x \rangle)$
 (* three place (x,y) *) $(\lambda F a . \langle F, x, y, a \rangle) (\lambda F a . \langle F, x, a, y \rangle)$
 $(\lambda F a . \langle F, a, x, y \rangle)$
 (* three place (y,x) *) $(\lambda F a . \langle F, y, x, a \rangle) (\lambda F a . \langle F, y, a, x \rangle)$
 $(\lambda F a . \langle F, a, y, x \rangle)$
 (* three place (x,x,y) *) $(\lambda F . \langle F, x, x, y \rangle) (\lambda F . \langle F, x, y, x \rangle) (\lambda F . \langle F, y, x, x \rangle)$
 (* three place (x,y,y) *) $(\lambda F . \langle F, x, y, y \rangle) (\lambda F . \langle F, y, x, y \rangle) (\lambda F . \langle F, y, y, x \rangle)$
 (* three place (x,x,x) *) $(\lambda F . \langle F, x, x, x \rangle)$
 (* three place (y,y,y) *) $(\lambda F . \langle F, y, y, y \rangle)$

lemma *IsPropositionalInXY-intro*[*IsPropositional-intros*]:

IsPropositionalInXY $(\lambda x y . \chi$

(* only x *)
 (* one place *) $(\lambda F . \langle F, x \rangle)$
 (* two place *) $(\lambda F . \langle F, x, x \rangle) (\lambda F a . \langle F, x, a \rangle) (\lambda F a . \langle F, a, x \rangle)$
 (* three place three x *) $(\lambda F . \langle F, x, x, x \rangle)$
 (* three place two x *) $(\lambda F a . \langle F, x, x, a \rangle) (\lambda F a . \langle F, x, a, x \rangle)$
 $(\lambda F a . \langle F, a, x, x \rangle)$
 (* three place one x *) $(\lambda F a b . \langle F, x, a, b \rangle) (\lambda F a b . \langle F, a, x, b \rangle)$
 $(\lambda F a b . \langle F, a, b, x \rangle)$
 (* only y *)
 (* one place *) $(\lambda F . \langle F, y \rangle)$
 (* two place *) $(\lambda F . \langle F, y, y \rangle) (\lambda F a . \langle F, y, a \rangle) (\lambda F a . \langle F, a, y \rangle)$
 (* three place three y *) $(\lambda F . \langle F, y, y, y \rangle)$
 (* three place two y *) $(\lambda F a . \langle F, y, y, a \rangle) (\lambda F a . \langle F, y, a, y \rangle)$
 $(\lambda F a . \langle F, a, y, y \rangle)$
 (* three place one y *) $(\lambda F a b . \langle F, y, a, b \rangle) (\lambda F a b . \langle F, a, y, b \rangle)$
 $(\lambda F a b . \langle F, a, b, y \rangle)$
 (* x and y *)
 (* two place *) $(\lambda F . \langle F, x, y \rangle) (\lambda F . \langle F, y, x \rangle)$
 (* three place (x,y) *) $(\lambda F a . \langle F, x, y, a \rangle) (\lambda F a . \langle F, x, a, y \rangle)$
 $(\lambda F a . \langle F, a, x, y \rangle)$
 (* three place (y,x) *) $(\lambda F a . \langle F, y, x, a \rangle) (\lambda F a . \langle F, y, a, x \rangle)$
 $(\lambda F a . \langle F, a, y, x \rangle)$
 (* three place (x,x,y) *) $(\lambda F . \langle F, x, x, y \rangle) (\lambda F . \langle F, x, y, x \rangle)$
 $(\lambda F . \langle F, y, x, x \rangle)$
 (* three place (x,y,y) *) $(\lambda F . \langle F, x, y, y \rangle) (\lambda F . \langle F, y, x, y \rangle)$
 $(\lambda F . \langle F, y, y, x \rangle)$
 (* three place (x,x,x) *) $(\lambda F . \langle F, x, x, x \rangle)$
 (* three place (y,y,y) *) $(\lambda F . \langle F, y, y, y \rangle)$

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* $*) (\lambda F . \langle F, x \rangle)$

$(*$ *two place* $*) (\lambda F . \langle F, x, x \rangle) (\lambda F a . \langle F, x, a \rangle) (\lambda F a . \langle F, a, x \rangle)$

$(*$ *three place three x* $*) (\lambda F . \langle F, x, x, x \rangle)$

$(*$ *three place two x* $*) (\lambda F a . \langle F, x, x, a \rangle) (\lambda F a . \langle F, x, a, x \rangle)$

$(\lambda F a . \langle F, a, x, x \rangle)$

$(*$ *three place one x* $*) (\lambda F a b . \langle F, x, a, b \rangle) (\lambda F a b . \langle F, a, x, b \rangle)$

$(\lambda F a b . \langle F, a, b, x \rangle)$

$(*$ *only y* $*)$

$(*$ *one place* $*) (\lambda F . \langle F, y \rangle)$

$(*$ *two place* $*) (\lambda F . \langle F, y, y \rangle) (\lambda F a . \langle F, y, a \rangle) (\lambda F a . \langle F, a, y \rangle)$

$(*$ *three place three y* $*) (\lambda F . \langle F, y, y, y \rangle)$

$(*$ *three place two y* $*) (\lambda F a . \langle F, y, y, a \rangle) (\lambda F a . \langle F, y, a, y \rangle)$

$(\lambda F a . \langle F, a, y, y \rangle)$

$(*$ *three place one y* $*) (\lambda F a b . \langle F, y, a, b \rangle) (\lambda F a b . \langle F, a, y, b \rangle)$

$(\lambda F a b . \langle F, a, b, y \rangle)$

$(*$ *only z* $*)$

$(*$ *one place* $*) (\lambda F . \langle F, z \rangle)$

$(*$ *two place* $*) (\lambda F . \langle F, z, z \rangle) (\lambda F a . \langle F, z, a \rangle) (\lambda F a . \langle F, a, z \rangle)$

$(*$ *three place three z* $*) (\lambda F . \langle F, z, z, z \rangle)$

$(*$ *three place two z* $*) (\lambda F a . \langle F, z, z, a \rangle) (\lambda F a . \langle F, z, a, z \rangle)$

$(\lambda F a . \langle F, a, z, z \rangle)$

$(*$ *three place one z* $*) (\lambda F a b . \langle F, z, a, b \rangle) (\lambda F a b . \langle F, a, z, b \rangle)$

$(\lambda F a b . \langle F, a, b, z \rangle)$

$(*$ *x and y* $*)$

$(*$ *two place* $*) (\lambda F . \langle F, x, y \rangle) (\lambda F . \langle F, y, x \rangle)$

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

$(\lambda F a . \langle F, a, x, y \rangle)$

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

$(\lambda F a . \langle F, a, y, x \rangle)$

$(*$ *three place (x,x,y)* $*) (\lambda F . \langle F, x, x, y \rangle) (\lambda F . \langle F, x, y, x \rangle)$

$(\lambda F . \langle F, y, x, x \rangle)$

$(*$ *three place (x,y,y)* $*) (\lambda F . \langle F, x, y, y \rangle) (\lambda F . \langle F, y, x, y \rangle)$

$(\lambda F . \langle F, y, y, x \rangle)$

$(*$ *three place (x,x,x)* $*) (\lambda F . \langle F, x, x, x \rangle)$

$(*$ *three place (y,y,y)* $*) (\lambda F . \langle F, y, y, y \rangle)$

$(*$ *x and z* $*)$

$(*$ *two place* $*) (\lambda F . \langle F, x, z \rangle) (\lambda F . \langle F, z, x \rangle)$

$(*$ *three place (x,z)* $*) (\lambda F a . \langle F, x, z, a \rangle) (\lambda F a . \langle F, x, a, z \rangle)$

$(\lambda F a . \langle F, a, x, z \rangle)$

$(*$ *three place (z,x)* $*) (\lambda F a . \langle F, z, x, a \rangle) (\lambda F a . \langle F, z, a, x \rangle)$

$(\lambda F a . \langle F, a, z, x \rangle)$

$(*$ *three place (x,x,z)* $*) (\lambda F . \langle F, x, x, z \rangle) (\lambda F . \langle F, x, z, x \rangle)$

$(\lambda F . \langle F, z, x, x \rangle)$

$(*$ *three place (x,z,z)* $*) (\lambda F . \langle F, x, z, z \rangle) (\lambda F . \langle F, z, x, z \rangle)$

$(\lambda F . \langle F, z, z, x \rangle)$

$(*$ *three place (x,x,x)* $*) (\lambda F . \langle F, x, x, x \rangle)$

$(*$ *three place (z,z,z)* $*) (\lambda F . \langle F, z, z, z \rangle)$

$(*$ *y and z* $*)$

$(*$ *two place* $*) (\lambda F . \langle F, y, z \rangle) (\lambda F . \langle F, z, y \rangle)$

$(*$ *three place (y,z)* $*) (\lambda F a . \langle F, y, z, a \rangle) (\lambda F a . \langle F, y, a, z \rangle)$

$(\lambda F a . \langle F, a, y, z \rangle)$

(* three place (z,y) *) (λ F a . (F,z,y,a)) (λ F a . (F,z,a,y))
 (λ F a . (F,a,z,y))
 (* three place (y,y,z) *) (λ F . (F,y,y,z)) (λ F . (F,y,z,y))
 (λ F . (F,z,y,y))
 (* three place (y,z,z) *) (λ F . (F,y,z,z)) (λ F . (F,z,y,z))
 (λ F . (F,z,z,y))
 (* three place (y,y,y) *) (λ F . (F,y,y,y))
 (* three place (z,z,z) *) (λ F . (F,z,z,z))
 (* x y z *)
 (* three place (x,...) *) (λ F . (F,x,y,z)) (λ F . (F,x,z,y))
 (* three place (y,...) *) (λ F . (F,y,x,z)) (λ F . (F,y,z,x))
 (* three place (z,...) *) (λ F . (F,z,x,y)) (λ F . (F,z,y,x))

lemma *IsPropositionalInXYZ-intro*[*IsPropositional-intros*]:

IsPropositionalInXYZ (λ x y z . χ
 (* only x *)
 (* one place *) (λ F . (F,x))
 (* two place *) (λ F . (F,x,x)) (λ F a . (F,x,a)) (λ F a . (F,a,x))
 (* three place three x *) (λ F . (F,x,x,x))
 (* three place two x *) (λ F a . (F,x,x,a)) (λ F a . (F,x,a,x))
 (λ F a . (F,a,x,x))
 (* three place one x *) (λ F a b . (F,x,a,b)) (λ F a b . (F,a,x,b))
 (λ F a b . (F,a,b,x))
 (* only y *)
 (* one place *) (λ F . (F,y))
 (* two place *) (λ F . (F,y,y)) (λ F a . (F,y,a)) (λ F a . (F,a,y))
 (* three place three y *) (λ F . (F,y,y,y))
 (* three place two y *) (λ F a . (F,y,y,a)) (λ F a . (F,y,a,y))
 (λ F a . (F,a,y,y))
 (* three place one y *) (λ F a b . (F,y,a,b)) (λ F a b . (F,a,y,b))
 (λ F a b . (F,a,b,y))
 (* only z *)
 (* one place *) (λ F . (F,z))
 (* two place *) (λ F . (F,z,z)) (λ F a . (F,z,a)) (λ F a . (F,a,z))
 (* three place three z *) (λ F . (F,z,z,z))
 (* three place two z *) (λ F a . (F,z,z,a)) (λ F a . (F,z,a,z))
 (λ F a . (F,a,z,z))
 (* three place one z *) (λ F a b . (F,z,a,b)) (λ F a b . (F,a,z,b))
 (λ F a b . (F,a,b,z))
 (* x and y *)
 (* two place *) (λ F . (F,x,y)) (λ F . (F,y,x))
 (* three place (x,y) *) (λ F a . (F,x,y,a)) (λ F a . (F,x,a,y))
 (λ F a . (F,a,x,y))
 (* three place (y,x) *) (λ F a . (F,y,x,a)) (λ F a . (F,y,a,x))
 (λ F a . (F,a,y,x))
 (* three place (x,x,y) *) (λ F . (F,x,x,y)) (λ F . (F,x,y,x))
 (λ F . (F,y,x,x))
 (* three place (x,y,y) *) (λ F . (F,x,y,y)) (λ F . (F,y,x,y))
 (λ F . (F,y,y,x))
 (* three place (x,x,x) *) (λ F . (F,x,x,x))
 (* three place (y,y,y) *) (λ F . (F,y,y,y))
 (* x and z *)
 (* two place *) (λ F . (F,x,z)) (λ F . (F,z,x))
 (* three place (x,z) *) (λ F a . (F,x,z,a)) (λ F a . (F,x,a,z))

```

      (λ F a . (⟦F, a, x, z⟧))
(* three place (z, x) *) (λ F a . (⟦F, z, x, a⟧)) (λ F a . (⟦F, z, a, x⟧))
      (λ F a . (⟦F, a, z, x⟧))
(* three place (x, x, z) *) (λ F . (⟦F, x, x, z⟧)) (λ F . (⟦F, x, z, x⟧))
      (λ F . (⟦F, z, x, x⟧))
(* three place (x, z, z) *) (λ F . (⟦F, x, z, z⟧)) (λ F . (⟦F, z, x, z⟧))
      (λ F . (⟦F, z, z, x⟧))
(* three place (x, x, x) *) (λ F . (⟦F, x, x, x⟧))
(* three place (z, z, z) *) (λ F . (⟦F, z, z, z⟧))
(* y and z *)
(* two place *) (λ F . (⟦F, y, z⟧)) (λ F . (⟦F, z, y⟧))
(* three place (y, z) *) (λ F a . (⟦F, y, z, a⟧)) (λ F a . (⟦F, y, a, z⟧))
      (λ F a . (⟦F, a, y, z⟧))
(* three place (z, y) *) (λ F a . (⟦F, z, y, a⟧)) (λ F a . (⟦F, z, a, y⟧))
      (λ F a . (⟦F, a, z, y⟧))
(* three place (y, y, z) *) (λ F . (⟦F, y, y, z⟧)) (λ F . (⟦F, y, z, y⟧))
      (λ F . (⟦F, z, y, y⟧))
(* three place (y, z, z) *) (λ F . (⟦F, y, z, z⟧)) (λ F . (⟦F, z, y, z⟧))
      (λ F . (⟦F, z, z, y⟧))
(* three place (y, y, y) *) (λ F . (⟦F, y, y, y⟧))
(* three place (z, z, z) *) (λ F . (⟦F, z, z, z⟧))
(* x y z *)
(* three place (x, ...) *) (λ F . (⟦F, x, y, z⟧)) (λ F . (⟦F, x, z, y⟧))
(* three place (y, ...) *) (λ F . (⟦F, y, x, z⟧)) (λ F . (⟦F, y, z, x⟧))
(* three place (z, ...) *) (λ F . (⟦F, z, x, y⟧)) (λ F . (⟦F, z, y, x⟧))
unfolding 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  $R_0 = j \Rightarrow i \Rightarrow \text{bool}$ 
type-synonym  $R_1 = v \Rightarrow R_0$ 
type-synonym  $R_2 = v \Rightarrow v \Rightarrow R_0$ 
type-synonym  $R_3 = v \Rightarrow v \Rightarrow v \Rightarrow R_0$ 
type-synonym  $W = i$ 

```

Denotations of the terms in the language.

```

lift-definition  $d_\kappa :: \kappa \Rightarrow R_\kappa$  option is
  λ x . (if fst x then Some (snd x) 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 $ex1 \equiv \lambda F \ w . \{ x . F \ (\nu \nu \ x) \ dj \ w \}$

definition $ex2 :: R_2 \Rightarrow W \Rightarrow ((R_\kappa \times R_\kappa) \ set)$

where $ex2 \equiv \lambda F \ w . \{ (x,y) . F \ (\nu \nu \ x) \ (\nu \nu \ y) \ dj \ w \}$

definition $ex3 :: R_3 \Rightarrow W \Rightarrow ((R_\kappa \times R_\kappa \times R_\kappa) \ set)$

where $ex3 \equiv \lambda F \ w . \{ (x,y,z) . F \ (\nu \nu \ x) \ (\nu \nu \ y) \ (\nu \nu \ z) \ dj \ w \}$

Encoding extensions.

definition $en :: R_1 \Rightarrow (R_\kappa \ set)$

where $en \equiv \lambda F . \{ x . \text{case } x \text{ of } \alpha \nu \ y \Rightarrow \text{make}\Pi_1 \ (\lambda x . F \ x) \in y \mid - \Rightarrow \text{False} \}$

Collect definitions.

named-theorems *semantics-defs*

declare $d_0\text{-def}[semantics-defs] \ d_1\text{-def}[semantics-defs]$

$d_2\text{-def}[semantics-defs] \ d_3\text{-def}[semantics-defs]$

$ex0\text{-def}[semantics-defs] \ ex1\text{-def}[semantics-defs]$

$ex2\text{-def}[semantics-defs] \ ex3\text{-def}[semantics-defs]$

$en\text{-def}[semantics-defs] \ d_\kappa\text{-def}[semantics-defs]$

$w_0\text{-def}[semantics-defs]$

Semantics for exemplification and encoding.

lemma $T1-1[semantics]$:

$(w \models \llbracket F, x \rrbracket) = (\exists \ r \ o_1 . \text{Some } r = d_1 \ F \wedge \text{Some } o_1 = d_\kappa \ x \wedge o_1 \in \text{ex1 } r \ w)$

unfolding *semantics-defs*

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

lemma $T1-2[semantics]$:

$(w \models \llbracket F, x, y \rrbracket) = (\exists \ r \ o_1 \ o_2 . \text{Some } r = d_2 \ F \wedge \text{Some } o_1 = d_\kappa \ x \wedge \text{Some } o_2 = d_\kappa \ y \wedge (o_1, o_2) \in \text{ex2 } r \ w)$

unfolding *semantics-defs*

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

lemma $T1-3[semantics]$:

$(w \models \llbracket F, x, y, z \rrbracket) = (\exists \ r \ o_1 \ o_2 \ o_3 . \text{Some } r = d_3 \ F \wedge \text{Some } o_1 = d_\kappa \ x \wedge \text{Some } o_2 = d_\kappa \ y \wedge \text{Some } o_3 = d_\kappa \ z \wedge (o_1, o_2, o_3) \in \text{ex3 } r \ w)$

unfolding *semantics-defs*

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

lemma $T2[semantics]$:

$(w \models \llbracket x, F \rrbracket) = (\exists \ r \ o_1 . \text{Some } r = d_1 \ F \wedge \text{Some } o_1 = d_\kappa \ x \wedge o_1 \in \text{en } r)$

unfolding *semantics-defs*

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

lemma *T3[semantics]*:
 $(w \models \llbracket F \rrbracket) = (\exists r . \text{Some } r = d_0 F \wedge \text{ex0 } r w)$
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]*: $(w \models \mathcal{A}\psi) = (dw \models \psi)$
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]*: $(w \models \forall_\circ x . \psi x) = (\forall x . (w \models \psi x))$
by (*simp add: meta-defs meta-aux*)

Semantics for descriptions and lambda expressions.

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

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

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

lemma *D4-3[semantics]*: $d_3 (\lambda^3 (\lambda x y z . \llbracket F, x^P, y^P, z^P \rrbracket)) = d_3 F$
by (*simp add: meta-defs meta-aux*)

lemma *D5-1[semantics]*:
assumes *IsPropositionalInX* φ
shows $\bigwedge w \ o_1 \ r . \text{Some } r = d_1 (\lambda x . (\varphi (x^P))) \wedge \text{Some } o_1 = d_\kappa x$
 $\longrightarrow (o_1 \in \text{ex1 } r \ w) = (w \models \varphi \ x)$
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 $\bigwedge w \ o_1 \ o_2 \ r . \text{Some } r = d_2 (\lambda^2 x \ y . \varphi (x^P) (y^P))$
 $\wedge \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y$
 $\longrightarrow ((o_1, o_2) \in \text{ex2 } r \ w) = (w \models \varphi \ x \ y)$
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 $\bigwedge w \ o_1 \ o_2 \ o_3 \ r . \text{Some } r = d_3 (\lambda^3 x \ y \ z . \varphi (x^P) (y^P) (z^P))$
 $\wedge \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge \text{Some } o_3 = d_\kappa z$
 $\longrightarrow ((o_1, o_2, o_3) \in \text{ex3 } r \ w) = (w \models \varphi \ x \ y \ z)$
using *assms unfolding IsPropositionalIn-defs semantics-defs*
by (*auto simp: meta-defs meta-aux denotes-def denotation-def*)

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

Auxiliary lemmata.

lemma *propex1*: $\exists r . \text{Some } r = d_1 F$
unfolding *d1-def* **by** *simp*
lemma *d1-inject*: $\bigwedge x \ y . d_1 x = d_1 y \implies x = y$
unfolding *d1-def* **by** (*simp add: evalΠ1-inject*)
lemma *dκ-inject*: $\bigwedge x \ y \ o_1 . \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_1 = d_\kappa y \implies x = y$
proof –
fix $x :: \kappa$ **and** $y :: \kappa$ **and** $o_1 :: \nu$
assume $\text{Some } o_1 = d_\kappa x \wedge \text{Some } o_1 = d_\kappa y$
moreover **hence**
 $\text{fst } (\text{eval}\kappa \ x) \wedge \text{fst } (\text{eval}\kappa \ y) \wedge \text{snd } (\text{eval}\kappa \ x) = o_1 \wedge \text{snd } (\text{eval}\kappa \ x) = o_1$
unfolding *dκ-def*
apply *transfer*
apply *simp*
by (*metis option.distinct(1) option.inject*)
ultimately **show** $x = y$
unfolding *dκ-def*
apply *transfer*
by *auto*
qed
lemma *dκ-proper*: $d_\kappa (u^P) = \text{Some } u$
unfolding *dκ-def* **by** (*simp add: νκ-def meta-aux*)
end

3.3 Validity Syntax

abbreviation *validity-in* :: $\text{o} \Rightarrow \text{i} \Rightarrow \text{bool}$ ($[- \text{ in } -] [1]$) **where**
 $\text{validity-in} \equiv \lambda \varphi \ v . v \models \varphi$
abbreviation *actual-validity* :: $\text{o} \Rightarrow \text{bool}$ ($[-] [1]$) **where**
 $\text{actual-validity} \equiv \lambda \varphi . \text{dw} \models \varphi$
abbreviation *necessary-validity* :: $\text{o} \Rightarrow \text{bool}$ ($\Box[-] [1]$) **where**
 $\text{necessary-validity} \equiv \lambda \varphi . \forall \ v . (v \models \varphi)$

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. Furthermore 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*]: $([\varphi \text{ in } v] \Rightarrow [\psi \text{ in } v]) \Rightarrow ([\varphi \rightarrow \psi \text{ in } v])$
by (*simp add: Semantics.T5*)

lemma *ImplE*[*meta-elim*]: $([\varphi \rightarrow \psi \text{ in } v]) \Rightarrow ([\varphi \text{ in } v] \longrightarrow [\psi \text{ in } v])$
by (*simp add: Semantics.T5*)

lemma *ImplS*[*meta-subst*]: $([\varphi \rightarrow \psi \text{ in } v]) = ([\varphi \text{ in } v] \longrightarrow [\psi \text{ in } v])$
by (*simp add: Semantics.T5*)

4.2 Rules for Negation

lemma *NotI*[*meta-intro*]: $\neg[\varphi \text{ in } v] \Rightarrow \neg[\varphi \text{ in } v]$
by (*simp add: Semantics.T4*)

lemma *NotE*[*meta-elim*]: $\neg[\varphi \text{ in } v] \Rightarrow \neg[\varphi \text{ in } v]$
by (*simp add: Semantics.T4*)

lemma *NotS*[*meta-subst*]: $\neg[\varphi \text{ in } v] = (\neg[\varphi \text{ in } v])$
by (*simp add: Semantics.T4*)

4.3 Rules for Conjunction

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

4.4 Rules for Equivalence

lemma *EquivI*[meta-intro]: $([\varphi \text{ in } v] \longleftrightarrow [\psi \text{ in } v]) \implies [\varphi \equiv \psi \text{ in } v]$
by (*simp add: equiv-def NotS ImplS ConjS*)
lemma *EquivE*[meta-elim]: $[\varphi \equiv \psi \text{ in } v] \implies ([\varphi \text{ in } v] \longleftrightarrow [\psi \text{ in } v])$
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]: $([\varphi \text{ in } v] \vee [\psi \text{ in } v]) \implies [\varphi \vee \psi \text{ in } v]$
by (*auto simp: disj-def NotS ImplS*)
lemma *DisjE*[meta-elim]: $[\varphi \vee \psi \text{ in } v] \implies ([\varphi \text{ in } v] \vee [\psi \text{ in } v])$
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]: $(\bigwedge v. [\varphi \text{ in } v]) \implies [\Box \varphi \text{ in } v]$
by (*simp add: Semantics.T6*)
lemma *BoxE*[meta-elim]: $[\Box \varphi \text{ in } v] \implies (\bigwedge v. [\varphi \text{ in } v])$
by (*simp add: Semantics.T6*)
lemma *BoxS*[meta-subst]: $[\Box \varphi \text{ in } v] = (\bigwedge v. [\varphi \text{ in } v])$
by (*simp add: Semantics.T6*)

4.7 Rules for Possibility

lemma *DiaI*[meta-intro]: $(\exists v. [\varphi \text{ in } v]) \implies [\Diamond \varphi \text{ in } v]$
by (*metis BoxS NotS diamond-def*)
lemma *DiaE*[meta-elim]: $[\Diamond \varphi \text{ in } v] \implies (\exists v. [\varphi \text{ in } v])$
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_νI*[meta-intro]: $(\bigwedge x::\nu. [\varphi \ x \text{ in } v]) \implies [\forall_{\nu} x. \varphi \ x \text{ in } v]$
by (*auto simp: Semantics.T8-ν*)
lemma *All_νE*[meta-elim]: $[\forall_{\nu} x. \varphi \ x \text{ in } v] \implies (\bigwedge x::\nu. [\varphi \ x \text{ in } v])$
by (*auto simp: Semantics.T8-ν*)
lemma *All_νS*[meta-subst]: $[\forall_{\nu} x. \varphi \ x \text{ in } v] = (\bigwedge x::\nu. [\varphi \ x \text{ in } v])$
by (*auto simp: Semantics.T8-ν*)
lemma *All₀I*[meta-intro]: $(\bigwedge x::\Pi_0. [\varphi \ x \text{ in } v]) \implies [\forall_0 x. \varphi \ x \text{ in } v]$

by (auto simp: Semantics.T8-0)
lemma All₀E[meta-elim]: $[\forall_0 x. \varphi \ x \text{ in } v] \implies (\bigwedge x::\Pi_0. [\varphi \ x \text{ in } v])$
 by (auto simp: Semantics.T8-0)
lemma All₀S[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 All₁I[meta-intro]: $(\bigwedge x::\Pi_1. [\varphi \ x \text{ in } v]) \implies [\forall_1 x. \varphi \ x \text{ in } v]$
 by (auto simp: Semantics.T8-1)
lemma All₁E[meta-elim]: $[\forall_1 x. \varphi \ x \text{ in } v] \implies (\bigwedge x::\Pi_1. [\varphi \ x \text{ in } v])$
 by (auto simp: Semantics.T8-1)
lemma All₁S[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 All₂I[meta-intro]: $(\bigwedge x::\Pi_2. [\varphi \ x \text{ in } v]) \implies [\forall_2 x. \varphi \ x \text{ in } v]$
 by (auto simp: Semantics.T8-2)
lemma All₂E[meta-elim]: $[\forall_2 x. \varphi \ x \text{ in } v] \implies (\bigwedge x::\Pi_2. [\varphi \ x \text{ in } v])$
 by (auto simp: Semantics.T8-2)
lemma All₂S[meta-subst]: $[\forall_2 x. \varphi \ x \text{ in } v] = (\forall x::\Pi_2. [\varphi \ x \text{ in } v])$
 by (auto simp: Semantics.T8-2)

lemma All₃I[meta-intro]: $(\bigwedge x::\Pi_3. [\varphi \ x \text{ in } v]) \implies [\forall_3 x. \varphi \ x \text{ in } v]$
 by (auto simp: Semantics.T8-3)
lemma All₃E[meta-elim]: $[\forall_3 x. \varphi \ x \text{ in } v] \implies (\bigwedge x::\Pi_3. [\varphi \ x \text{ in } v])$
 by (auto simp: Semantics.T8-3)
lemma All₃S[meta-subst]: $[\forall_3 x. \varphi \ x \text{ in } v] = (\forall x::\Pi_3. [\varphi \ x \text{ in } v])$
 by (auto simp: Semantics.T8-3)

4.9 Rules for Actuality

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

4.10 Rules for Encoding

lemma EncI[meta-intro]:
 assumes $\exists r \ o_1. \text{Some } r = d_1 \ F \wedge \text{Some } o_1 = d_\kappa \ x \wedge o_1 \in \text{en } r$
 shows $[\llbracket x, F \rrbracket \text{ in } v]$
 using assms by (auto simp: Semantics.T2)

lemma EncE[meta-elim]:
 assumes $[\llbracket x, F \rrbracket \text{ in } v]$
 shows $\exists r \ o_1. \text{Some } r = d_1 \ F \wedge \text{Some } o_1 = d_\kappa \ x \wedge o_1 \in \text{en } r$
 using assms by (auto simp: Semantics.T2)

lemma EncS[meta-subst]:
 $[\llbracket x, F \rrbracket \text{ in } v] = (\exists r \ o_1. \text{Some } r = d_1 \ F \wedge \text{Some } o_1 = d_\kappa \ x \wedge o_1 \in \text{en } r)$
 by (auto simp: Semantics.T2)

4.11 Rules for Exemplification

4.11.1 Zero-place Relations

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

4.11.2 One-Place Relations

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

4.11.3 Two-Place Relations

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

4.11.4 Three-Place Relations

lemma *Exe3I*[*meta-intro*]:
assumes $\exists r o_1 o_2 o_3 . \text{Some } r = d_3 F \wedge \text{Some } o_1 = d_\kappa x$
 $\wedge \text{Some } o_2 = d_\kappa y \wedge \text{Some } o_3 = d_\kappa z$
 $\wedge (o_1, o_2, o_3) \in \text{ex3 } r v$
shows $[\langle F, x, y, z \rangle \text{ in } v]$
using *assms* **by** (*auto simp: Semantics.T1-3*)

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

4.12 Rules for Being Ordinary

lemma *OrdI*[*meta-intro*]:
assumes $\exists o_1 \ y. \text{Some } o_1 = d_\kappa \ x \wedge o_1 = \omega\nu \ y$
shows $[\![O!, x]\!] \text{ in } v$
proof –
obtain o_1 **and** y **where** $1: \text{Some } o_1 = d_\kappa \ x \wedge o_1 = \omega\nu \ y$
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) vv- $\omega\nu$ -is- $\omega\nu$ v.simps(5)*
 $\text{option.distinct}(1) \text{ option.sel}$)
qed
lemma *OrdE*[*meta-elim*]:
assumes $[\![O!, x]\!] \text{ in } v$
shows $\exists o_1 \ y. \text{Some } o_1 = d_\kappa \ x \wedge o_1 = \omega\nu \ y$
using *assms* **unfolding** *Ordinary-def conn-defs meta-defs*
apply (*simp add: meta-aux d $_\kappa$ -def denotes-def denotation-def*)
by (*metis v.exhaust v.simps(6) vv-def v.simps(6) comp-apply*)
lemma *OrdS*[*meta-cong*]:
 $[\![O!, x]\!] \text{ in } v = (\exists o_1 \ y. \text{Some } o_1 = d_\kappa \ x \wedge o_1 = \omega\nu \ y)$
using *OrdI OrdE* **by** *blast*

4.13 Rules for Being Abstract

lemma *AbsI*[*meta-intro*]:
assumes $\exists o_1 \ y. \text{Some } o_1 = d_\kappa \ x \wedge o_1 = \alpha\nu \ y$
shows $[\![A!, x]\!] \text{ in } v$
proof –
obtain $o_1 \ y$ **where** $\text{Some } o_1 = d_\kappa \ x \wedge o_1 = \alpha\nu \ y$
using *assms* **by** *auto*
thus *?thesis*
unfolding *Abstract-def conn-defs meta-defs*
apply (*simp add: meta-aux*)
by (*metis d $_\kappa$ -inject d $_\kappa$ -proper v.simps(6) vv-def v.simps(6)*
 $\text{o-apply proper-denotation proper-denotes}$)
qed
lemma *AbsE*[*meta-elim*]:

```

assumes  $[(A!,x) \text{ in } v]$ 
shows  $\exists o_1 y. \text{Some } o_1 = d_\kappa x \wedge o_1 = \alpha\nu y$ 
using assms unfolding conn-defs meta-defs Abstract-def
apply (simp add: meta-aux d $\kappa$ -def denotes-def denotation-def)
by (metis OrdinaryObjectsPossiblyConcreteAxiom  $\nu$ .exhaust
       $\nu\nu$ - $\omega\nu$ -is- $\omega\nu$   $v$ .simps(5)))
lemma AbsS[meta-cong]:
 $[(A!,x) \text{ in } v] = (\exists o_1 y. \text{Some } o_1 = d_\kappa x \wedge o_1 = \alpha\nu y)$ 
using AbsI AbsE by blast

```

4.14 Rules for Definite Descriptions

```

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

```

4.15 Rules for Identity

4.15.1 Ordinary Objects

```

lemma EqEI[meta-intro]:
  assumes  $\exists o_1 X o_2. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge o_1 = o_2$ 
   $\wedge o_1 = \omega\nu X$ 
  shows  $[x =_E y \text{ in } v]$ 
  using assms
  apply (simp add: meta-defs meta-aux basic-identityE-def basic-identityE-infix-def
        conn-defs Ordinary-def OrdinaryObjectsPossiblyConcreteAxiom
        denotes-def Semantics.d $\kappa$ -def
        split:  $\nu$ .split  $v$ .split)
  using OrdinaryObjectsPossiblyConcreteAxiom
  apply transfer
  apply simp
  by (metis  $\nu\nu$ - $\omega\nu$ -is- $\omega\nu$   $v$ .distinct(1)  $v$ .inject(1) option.distinct(1)
      option.sel)
lemma EqEE[meta-elim]:
  assumes  $[x =_E y \text{ in } v]$ 
  shows  $\exists o_1 X o_2. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge o_1 = o_2 \wedge$ 
   $o_1 = \omega\nu X$ 
proof –
  have  $1: [(O!,x) \ \& \ (O!,y) \ \& \ \Box(\forall_1 F. (F,x) \equiv (F,y)) \text{ in } v]$ 
  using assms unfolding basic-identityE-def basic-identityE-infix-def
  using D4-2 T1-2 D5-2 IsPropositional-intros by meson
  hence  $2: \exists o_1 o_2 X Y. \text{Some } o_1 = d_\kappa x \wedge o_1 = \omega\nu X$ 
       $\wedge \text{Some } o_2 = d_\kappa y \wedge o_2 = \omega\nu Y$ 
  apply (subst (asm) ConjS)
  apply (subst (asm) ConjS)
  using OrdE by auto
then obtain  $o_1 o_2 X Y$  where  $3:$ 
       $\text{Some } o_1 = d_\kappa x \wedge o_1 = \omega\nu X \wedge \text{Some } o_2 = d_\kappa y \wedge o_2 = \omega\nu Y$ 
  by auto
  have  $\exists r. \text{Some } r = d_1 (\lambda z. \text{makeo } (\lambda w s. d_\kappa (z^P) = \text{Some } o_1))$ 
  using propex1 by auto
then obtain  $r$  where  $4:$ 
       $\text{Some } r = d_1 (\lambda z. \text{makeo } (\lambda w s. d_\kappa (z^P) = \text{Some } o_1))$ 
  by auto

```

```

hence 5:  $r = (\lambda u \ w \ s. \text{Some } (v \nu \ u) = \text{Some } o_1)$ 
unfolding lambdabinder1-def d1-def dκ-proper
apply transfer
by simp
have  $\Box(\forall_1 F. \langle F, x \rangle \equiv \langle F, y \rangle) \text{ in } v$ 
using 1 using ConjE by blast
hence 6:  $\forall v \ F. [\langle F, x \rangle \text{ in } v] \longleftrightarrow [\langle F, y \rangle \text{ in } v]$ 
using BoxE EquivE All1E by fast
hence 7:  $\forall v. (o_1 \in \text{ex1 } r \ v) = (o_2 \in \text{ex1 } r \ v)$ 
using 2 4 unfolding valid-in-def
by (metis 3 6 d1.rep-eq dκ-inject dκ-proper ex1-def evalo-inverse
ex1.rep-eq
mem-Collect-eq option.sel proper-denotation proper-denotes
valid-in.abs-eq)
have  $o_1 \in \text{ex1 } r \ v$ 
using 5 3 unfolding ex1-def by (simp add: meta-aux)
hence  $o_2 \in \text{ex1 } r \ v$ 
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 EqES[meta-subst]:
 $[x =_E y \text{ in } v] = (\exists o_1 \ X \ o_2. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y$ 
 $\wedge o_1 = o_2 \wedge o_1 = \omega \nu \ X)$ 
using EqEI EqEE by blast

```

4.15.2 Individuals

```

lemma EqκI[meta-intro]:
assumes  $\exists o_1 \ o_2. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge o_1 = o_2$ 
shows  $[x =_\kappa y \text{ in } v]$ 
proof —
have  $x = y$  using assms dκ-inject by meson
moreover have  $[x =_\kappa x \text{ in } v]$ 
unfolding basic-identityκ-def
apply meta-solver
by (metis (no-types, lifting) assms AbsI Exe1E ν.exhaust)
ultimately show ?thesis by auto
qed
lemma Eqκ-prop:
assumes  $[x =_\kappa y \text{ in } v]$ 
shows  $[\varphi \ x \text{ in } v] = [\varphi \ y \text{ in } v]$ 
proof —
have  $[x =_E y \vee \langle A!, x \rangle \ \& \ \langle A!, y \rangle \ \& \ \Box(\forall_1 F. \langle x, F \rangle \equiv \langle y, F \rangle) \text{ in } v]$ 
using assms unfolding basic-identityκ-def by simp
moreover {
assume  $[x =_E y \text{ in } v]$ 
hence  $(\exists o_1 \ o_2. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge o_1 = o_2)$ 
using EqEE by fast
}
moreover {
assume  $! : [\langle A!, x \rangle \ \& \ \langle A!, y \rangle \ \& \ \Box(\forall_1 F. \langle x, F \rangle \equiv \langle y, F \rangle) \text{ in } v]$ 

```

hence 2: $(\exists o_1 o_2 X Y. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y$
 $\quad \wedge o_1 = \alpha\nu X \wedge o_2 = \alpha\nu Y)$
 using *AbsE ConjE* by *meson*
 moreover then obtain $o_1 o_2 X Y$ where 3:
 $\text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge o_1 = \alpha\nu X \wedge o_2 = \alpha\nu Y$
 by *auto*
 moreover have 4: $[\Box(\forall_1 F. \llbracket x, F \rrbracket \equiv \llbracket y, F \rrbracket) \text{ in } v]$
 using *1 ConjE* by *blast*
 hence 6: $\forall v F. [\llbracket x, F \rrbracket \text{ in } v] \longleftrightarrow [\llbracket y, F \rrbracket \text{ in } v]$
 using *BoxE All₁E EquivE* by *fast*
 hence 7: $\forall v r. (\exists o_1. \text{Some } o_1 = d_\kappa x \wedge o_1 \in \text{en } r)$
 $\quad = (\exists o_1. \text{Some } o_1 = d_\kappa y \wedge o_1 \in \text{en } r)$
 apply *cut-tac* apply *meta-solver*
 using *proper_{x1} d₁-inject* apply *simp*
 apply *transfer* by *simp*
 hence 8: $\forall r. (o_1 \in \text{en } r) = (o_2 \in \text{en } r)$
 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)*
 $\quad \text{mem-Collect-eq make}\Pi_1\text{-cases})$
 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 $(\exists o_1 o_2. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge o_1 = o_2)$
 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 *EqκE[meta-elim]*:
 assumes $[x =_\kappa y \text{ in } v]$
 shows $\exists o_1 o_2. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge o_1 = o_2$
 proof –
 have $\forall \varphi. (v \models \varphi x) = (v \models \varphi y)$
 using *assms Eqκ-prop* by *blast*
 moreover obtain φ where *φ-prop*:
 $\varphi = (\lambda \alpha. \text{makeo } (\lambda w s. (\exists o_1 o_2. \text{Some } o_1 = d_\kappa x$
 $\quad \wedge \text{Some } o_2 = d_\kappa \alpha \wedge o_1 = o_2)))$
 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_κ-def*
 by *(metis (mono-tags, lifting) AbsS ConjE DisjS*
 $\quad \text{Eq}_E S \text{ valid-in.abs-eq})$
 ultimately have $(v \models \varphi y)$ by *auto*
 thus *?thesis*
 unfolding *φ-prop*
 by *(simp add: valid-in-def meta-aux)*
 qed
 lemma *EqκS[meta-subst]*:
 $[x =_\kappa y \text{ in } v] = (\exists o_1 o_2. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge o_1$

$= o_2)$
using $Eq\kappa I$ $Eq\kappa E$ **by** *blast*

4.15.3 One-Place Relations

lemma $Eq_1 I[meta-intro]$: $F = G \implies [F =_1 G \text{ in } v]$
unfolding *basic-identity₁-def*
apply (*rule* $BoxI$, *rule* $All_\nu I$, *rule* $EquivI$)
by *simp*
lemma $Eq_1 E[meta-elim]$: $[F =_1 G \text{ in } v] \implies F = G$
unfolding *basic-identity₁-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 $Eq_1 S[meta-subst]$: $[F =_1 G \text{ in } v] = (F = G)$
using $Eq_1 I$ $Eq_1 E$ **by** *auto*
lemma Eq_1 -*prop*: $[F =_1 G \text{ in } v] \implies [\varphi F \text{ in } v] = [\varphi G \text{ in } v]$
using $Eq_1 E$ **by** *blast*

4.15.4 Two-Place Relations

lemma $Eq_2 I[meta-intro]$: $F = G \implies [F =_2 G \text{ in } v]$
unfolding *basic-identity₂-def*
apply (*rule* $All_\nu I$, *rule* $ConjI$, (*subst* $Eq_1 S$)+)
by *simp*
lemma $Eq_2 E[meta-elim]$: $[F =_2 G \text{ in } v] \implies F = G$
proof –
assume $[F =_2 G \text{ in } v]$
hence $[\forall_\nu x. (\lambda y. \langle F, x^P, y^P \rangle) =_1 (\lambda y. \langle G, x^P, y^P \rangle) \text{ in } v]$
unfolding *basic-identity₂-def*
apply *cut-tac* **apply** *meta-solver* **by** *auto*
hence $\bigwedge x. (make\Pi_1 (eval\Pi_2 F (\nu\nu x)) = make\Pi_1 ((eval\Pi_2 G (\nu\nu x))))$
apply *cut-tac* **apply** *meta-solver*
by (*simp add: meta-defs meta-aux*)
hence $\bigwedge x. (eval\Pi_2 F (\nu\nu x) = eval\Pi_2 G (\nu\nu x))$
by (*simp add: make\Pi₁-inject*)
hence $\bigwedge x1. (eval\Pi_2 F x1 = eval\Pi_2 G x1)$
using $\nu\nu$ -*surj* **by** (*metis* $\nu\nu$ - $\nu\nu$ -*id*)
thus $F = G$ **using** *eval\Pi₂-inject* **by** *blast*
qed
lemma $Eq_2 S[meta-subst]$: $[F =_2 G \text{ in } v] = (F = G)$
using $Eq_2 I$ $Eq_2 E$ **by** *auto*
lemma Eq_2 -*prop*: $[F =_2 G \text{ in } v] \implies [\varphi F \text{ in } v] = [\varphi G \text{ in } v]$
using $Eq_2 E$ **by** *blast*

4.15.5 Three-Place Relations

lemma $Eq_3 I[meta-intro]$: $F = G \implies [F =_3 G \text{ in } v]$
apply (*simp add: meta-defs meta-aux conn-defs basic-identity₃-def*)
using *MetaSolver.Eq₁I valid-in.rep-eq* **by** *auto*
lemma $Eq_3 E[meta-elim]$: $[F =_3 G \text{ in } v] \implies F = G$
proof –

```

assume  $[F =_3 G \text{ in } v]$ 
hence  $[\forall_\nu x y. (\lambda z. \langle F, x^P, y^P, z^P \rangle) =_1 (\lambda z. \langle G, x^P, y^P, z^P \rangle) \text{ in } v]$ 
  unfolding basic-identity3-def apply cut-tac
  apply meta-solver by auto
hence  $\bigwedge x y. (\lambda z. \langle F, x^P, y^P, z^P \rangle) = (\lambda z. \langle G, x^P, y^P, z^P \rangle)$ 
  using Eq1E AllνS by (metis (mono-tags, lifting))
hence  $\bigwedge x y. \text{make}\Pi_1 (\text{eval}\Pi_3 F x y) = \text{make}\Pi_1 (\text{eval}\Pi_3 G x y)$ 
  apply (auto simp: meta-defs meta-aux)
  using νν-surj by (metis νν-νν-id)
thus  $F = G$  using make $\Pi_1$ -inject eval $\Pi_3$ -inject by blast
qed
lemma Eq3S[meta-subst]:  $[F =_3 G \text{ in } v] = (F = G)$ 
  using Eq3I Eq3E by auto
lemma Eq3-prop:  $[F =_3 G \text{ in } v] \implies [\varphi F \text{ in } v] = [\varphi G \text{ in } v]$ 
  using Eq3E by blast

```

4.15.6 Propositions

```

lemma EqoI[meta-intro]:  $x = y \implies [x =_o y \text{ in } v]$ 
  unfolding basic-identityo-def by (simp add: Eq1S)
lemma EqoE[meta-elim]:  $[F =_o G \text{ in } v] \implies F = G$ 
  unfolding basic-identityo-def
  apply (drule Eq1E)
  apply (simp add: meta-defs)
  using evalo-inject make $\Pi_1$ -inject
  by (metis UNIV-I)
lemma EqoS[meta-subst]:  $[F =_o G \text{ in } v] = (F = G)$ 
  using EqoI EqoE by auto
lemma Eqo-prop:  $[F =_o G \text{ in } v] \implies [\varphi F \text{ in } v] = [\varphi G \text{ in } v]$ 
  using EqoE 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 = νvar (varν:  $\nu$ ) | ovar (varo:  $o$ ) |  $\Pi_1$  var (var $\Pi_1$ :  $\Pi_1$ )
  |  $\Pi_2$  var (var $\Pi_2$ :  $\Pi_2$ ) |  $\Pi_3$  var (var $\Pi_3$ :  $\Pi_3$ )

```

Type class for quantifiable types:

```

class quantifiable = fixes forall :: ( $'a \Rightarrow o$ )  $\Rightarrow o$  (binder  $\forall$  [8] 9)
  and qvar ::  $'a \Rightarrow \text{var}$ 
  and varq ::  $\text{var} \Rightarrow 'a$ 

```

```

assumes quantifiable-T8:  $(w \models (\forall x. \psi x)) = (\forall x. (w \models (\psi x)))$ 
and varq-qvar-id:  $\text{varq } (qvar x) = x$ 
begin
  definition exists ::  $(\alpha \Rightarrow \text{bool}) \Rightarrow \text{bool}$  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  $\nu :: \text{quantifiable}$ 
begin
  definition forall- $\nu :: (\nu \Rightarrow \text{bool}) \Rightarrow \text{bool}$  where forall- $\nu \equiv \text{forall}_\nu$ 
  definition qvar- $\nu :: \nu \Rightarrow \text{var}$  where qvar  $\equiv \nu \text{var}$ 
  definition varq- $\nu :: \text{var} \Rightarrow \nu$  where varq  $\equiv \text{var} \nu$ 
  instance proof
    fix  $w :: i$  and  $\psi :: \nu \Rightarrow \text{bool}$ 
    show  $(w \models \forall x. \psi x) = (\forall x. (w \models \psi x))$ 
      unfolding forall- $\nu$ -def using Semantics.T8- $\nu$  .
  next
    fix  $x :: \nu$ 
    show  $\text{varq } (qvar x) = x$ 
      unfolding qvar- $\nu$ -def varq- $\nu$ -def by simp
  qed
end

```

```

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

```

```

instantiation  $\Pi_1 :: \text{quantifiable}$ 
begin
  definition forall- $\Pi_1 :: (\Pi_1 \Rightarrow \text{bool}) \Rightarrow \text{bool}$  where forall- $\Pi_1 \equiv \text{forall}_1$ 
  definition qvar- $\Pi_1 :: \Pi_1 \Rightarrow \text{var}$  where qvar  $\equiv \Pi_1 \text{var}$ 
  definition varq- $\Pi_1 :: \text{var} \Rightarrow \Pi_1$  where varq  $\equiv \text{var} \Pi_1$ 
  instance proof
    fix  $w :: i$  and  $\psi :: \Pi_1 \Rightarrow \text{bool}$ 

```

```

    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 ≡ Π2 var
  definition varq-Π2 :: var ⇒ Π2 where varq ≡ var Π2
  instance proof
    fix w :: i and ψ :: Π2 ⇒ o
    show (w ⊨ ∀ x. ψ x) = (∀ x. (w ⊨ ψ x))
      unfolding forall-Π2-def using Semantics.T8-2 .
  next
    fix x :: Π2
    show varq (qvar x) = x
      unfolding qvar-Π2-def varq-Π2-def by simp
  qed
end

instantiation Π3 :: quantifiable
begin
  definition forall-Π3 :: (Π3 ⇒ o) ⇒ o where forall-Π3 ≡ forall3
  definition qvar-Π3 :: Π3 ⇒ var where qvar ≡ Π3 var
  definition varq-Π3 :: var ⇒ Π3 where varq ≡ var Π3
  instance proof
    fix w :: i and ψ :: Π3 ⇒ o
    show (w ⊨ ∀ x. ψ x) = (∀ x. (w ⊨ ψ x))
      unfolding forall-Π3-def using Semantics.T8-3 .
  next
    fix x :: Π3
    show varq (qvar x) = x
      unfolding qvar-Π3-def varq-Π3-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]: (Λx::'a::quantifiable. [φ x in v]) ⇒ [∀ x. φ
x in v]
    by (auto simp: Semantics.T8)

```

```

lemma Alle[meta-elim]:  $[\forall x. \varphi x \text{ in } v] \implies (\bigwedge x::'a::\text{quantifiable}. [\varphi x \text{ in } v])$ 
by (auto simp: Semantics.T8)
lemma AllS[meta-subst]:  $[\forall x. \varphi x \text{ in } v] = (\forall x::'a::\text{quantifiable}. [\varphi x \text{ in } v])$ 
by (auto simp: Semantics.T8)

```

5.3.2 Rules for Existence

```

lemma ExIRule:  $([\varphi y \text{ in } v]) \implies [\exists x. \varphi x \text{ in } v]$ 
by (auto simp: exists-def NotS AllS)
lemma ExI[meta-intro]:  $(\exists y. [\varphi y \text{ in } v]) \implies [\exists x. \varphi x \text{ in } v]$ 
by (auto simp: exists-def NotS AllS)
lemma ExE[meta-elim]:  $[\exists x. \varphi x \text{ in } v] \implies (\exists y. [\varphi y \text{ in } v])$ 
by (auto simp: exists-def NotS AllS)
lemma ExS[meta-subst]:  $[\exists x. \varphi x \text{ in } v] = (\exists y. [\varphi y \text{ in } v])$ 
by (auto simp: exists-def NotS AllS)
lemma ExERule: assumes  $[\exists x. \varphi x \text{ in } v]$  obtains  $x$  where  $[\varphi x \text{ in } v]$ 

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:
 $w \models x = y \implies w \models \varphi x \implies w \models \varphi y$ 
begin
abbreviation notequal (infixl  $\neq$  63) where
 $notequal \equiv \lambda x y. \neg(x = y)$ 
end

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

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

```

6.2 Instantiations

```

instantiation  $\kappa$  :: identifiable

```

```

begin
  definition identity-κ where identity-κ  $\equiv$  basic-identityκ
  instance proof
    fix  $x\ y :: \kappa$  and  $w\ \varphi$ 
    show  $[x = y\ \text{in}\ w] \implies [\varphi\ x\ \text{in}\ w] \implies [\varphi\ y\ \text{in}\ w]$ 
      unfolding identity-κ-def
      using MetaSolver.Eqκ-prop ..
    qed
  end

instantiation  $\nu :: \text{identifiable}$ 
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  $\bigwedge \varphi. (v \models \varphi\ (\alpha^P)) \implies (v \models \varphi\ (\beta^P))$ 
      using l-identity by auto
    hence  $(v \models \varphi\ (\text{denotation}\ (\alpha^P))) \implies (v \models \varphi\ (\text{denotation}\ (\beta^P)))$ 
      by meson
    thus  $(v \models \varphi\ \alpha) \implies (v \models \varphi\ \beta)$ 
      by (simp only: proper-denotation)
    qed
  end

instantiation  $\Pi_1 :: \text{identifiable}$ 
begin
  definition identity-Π1 where identity-Π1  $\equiv$  basic-identity1
  instance proof
    fix  $F\ G :: \Pi_1$  and  $w\ \varphi$ 
    show  $(w \models F = G) \implies (w \models \varphi\ F) \implies (w \models \varphi\ G)$ 
      unfolding identity-Π1-def using MetaSolver.Eq1-prop ..
    qed
  end

instantiation  $\Pi_2 :: \text{identifiable}$ 
begin
  definition identity-Π2 where identity-Π2  $\equiv$  basic-identity2
  instance proof
    fix  $F\ G :: \Pi_2$  and  $w\ \varphi$ 
    show  $(w \models F = G) \implies (w \models \varphi\ F) \implies (w \models \varphi\ G)$ 
      unfolding identity-Π2-def using MetaSolver.Eq2-prop ..
    qed
  end

instantiation  $\Pi_3 :: \text{identifiable}$ 
begin
  definition identity-Π3 where identity-Π3  $\equiv$  basic-identity3
  instance proof
    fix  $F\ G :: \Pi_3$  and  $w\ \varphi$ 
    show  $(w \models F = G) \implies (w \models \varphi\ F) \implies (w \models \varphi\ G)$ 
      unfolding identity-Π3-def using MetaSolver.Eq3-prop ..

```

```

    qed
end

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

instance  $\nu$  :: quantifiable-and-identifiable ..
instance  $\Pi_1$  :: quantifiable-and-identifiable ..
instance  $\Pi_2$  :: quantifiable-and-identifiable ..
instance  $\Pi_3$  :: 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 identityE-def[identity-defs]:
  basic-identityE  $\equiv$   $\lambda^2$  ( $\lambda x y. (\downarrow O!, x^P) \ \& \ (\downarrow O!, y^P) \ \& \ \Box(\forall F. (\downarrow F, x^P) \equiv (\downarrow F, y^P)))$ )
  unfolding basic-identityE-def forall- $\Pi_1$ -def by simp
lemma identityE-infix-def[identity-defs]:
   $x =_E y \equiv (\downarrow \text{basic-identity}_E, x, y)$  using basic-identityE-infix-def .
lemma identity $\kappa$ -def[identity-defs]:
   $op = \equiv \lambda x y. x =_E y \vee (\downarrow A!, x) \ \& \ (\downarrow A!, y) \ \& \ \Box(\forall F. \llbracket x, F \rrbracket \equiv \llbracket y, F \rrbracket)$ 
  unfolding identity- $\kappa$ -def basic-identity $\kappa$ -def forall- $\Pi_1$ -def by simp
lemma identity $\nu$ -def[identity-defs]:
   $op = \equiv \lambda x y. (x^P) =_E (y^P) \vee (\downarrow A!, x^P) \ \& \ (\downarrow A!, y^P) \ \& \ \Box(\forall F. \llbracket x^P, F \rrbracket \equiv \llbracket y^P, F \rrbracket)$ 
  unfolding identity- $\nu$ -def identity- $\kappa$ -def by simp
lemma identity1-def[identity-defs]:
   $op = \equiv \lambda F G. \Box(\forall x. \llbracket x^P, F \rrbracket \equiv \llbracket x^P, G \rrbracket)$ 
  unfolding identity- $\Pi_1$ -def basic-identity1-def forall- $\nu$ -def by simp
lemma identity2-def[identity-defs]:
   $op = \equiv \lambda F G. \forall x. (\lambda y. (\downarrow F, x^P, y^P)) = (\lambda y. (\downarrow G, x^P, y^P))$ 
   $\ \& \ (\lambda y. (\downarrow F, y^P, x^P)) = (\lambda y. (\downarrow G, y^P, x^P))$ 
  unfolding identity- $\Pi_2$ -def identity- $\Pi_1$ -def basic-identity2-def forall- $\nu$ -def by simp
lemma identity3-def[identity-defs]:
   $op = \equiv \lambda F G. \forall x y. (\lambda z. (\downarrow F, z^P, x^P, y^P)) = (\lambda z. (\downarrow G, z^P, x^P, y^P))$ 
   $\ \& \ (\lambda z. (\downarrow F, x^P, z^P, y^P)) = (\lambda z. (\downarrow G, x^P, z^P, y^P))$ 
   $\ \& \ (\lambda z. (\downarrow F, x^P, y^P, z^P)) = (\lambda z. (\downarrow G, x^P, y^P, z^P))$ 
  unfolding identity- $\Pi_3$ -def identity- $\Pi_1$ -def basic-identity3-def forall- $\nu$ -def by simp
lemma identityo-def[identity-defs]:  $op = \equiv \lambda F G. (\lambda y. F) = (\lambda y. G)$ 

```

unfolding identity-o-def identity- Π_1 -def basic-identity_o-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 $[[\cdot]]$ 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 :: $\text{o} \Rightarrow \text{bool}$ ($[[\cdot]]$) **where** axiom $\equiv \lambda \varphi . \forall v . [\varphi \text{ in } v]$

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

lemma axiom-instance[axiom]: $[[\varphi]] \Rightarrow [\varphi \text{ in } v]$

unfolding axiom-def by simp

lemma closures-universal[axiom]: $(\bigwedge x. [[\varphi x]]) \Rightarrow [[\forall x. \varphi x]]$

by axiom-meta-solver

lemma closures-actualization[axiom]: $[[\varphi]] \Rightarrow [[\mathcal{A} \varphi]]$

by axiom-meta-solver

lemma closures-necessitation[axiom]: $[[\varphi]] \Rightarrow [[\Box \varphi]]$

by axiom-meta-solver

lemma necessitation-averse-axiom-instance[axiom]: $[\varphi] \Rightarrow [\varphi \text{ in } dw]$

by meta-solver

lemma necessitation-averse-closures-universal[axiom]: $(\bigwedge x. [\varphi x]) \Rightarrow [\forall x. \varphi x]$

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

```

7.3 Axioms of Identity

```

lemma l-identity[axiom]:
  [[ $\alpha = \beta \rightarrow (\varphi \alpha \rightarrow \varphi \beta)$ ]]
  using l-identity apply 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]:
  [[(∀ α. φ α) → φ α]]
  by axiom-meta-solver
lemma cqt-1-κ[axiom]:
  [[(∀ α. φ (αP)) → ((∃ β . (βP) = α) → φ α)]]
  proof –
  {
    fix v
    assume 1: [(∀ α. φ (αP)) in v]
    assume [(∃ β . (βP) = α) in v]
    then obtain β where 2:
      [(βP) = α in v] by (rule ExERule)
    hence [φ (βP) in v] using 1 Alle by blast
    hence [φ α in v]
      using l-identity[where φ=φ, axiom-instance]
      ImplS 2 by simp
  }
  thus [[(∀ α. φ (αP)) → ((∃ β . (βP) = α) → φ α)]]
  unfolding axiom-def using ImplI by blast
qed
lemma cqt-3[axiom]:
  [[(∀ α. φ α → ψ α) → ((∀ α. φ α) → (∀ α. ψ α))]]
  by axiom-meta-solver
lemma cqt-4[axiom]:
  [[φ → (∀ α. φ)]]
  by axiom-meta-solver

inductive SimpleExOrEnc
  where SimpleExOrEnc (λ x . ⟨F, x⟩)
    | SimpleExOrEnc (λ x . ⟨F, x, y⟩)
    | SimpleExOrEnc (λ x . ⟨F, y, x⟩)
    | SimpleExOrEnc (λ x . ⟨F, x, y, z⟩)
    | SimpleExOrEnc (λ x . ⟨F, y, x, z⟩)
    | SimpleExOrEnc (λ x . ⟨F, y, z, x⟩)
    | SimpleExOrEnc (λ x . ⟨x, F⟩)

lemma cqt-5[axiom]:
  assumes SimpleExOrEnc ψ
  shows [[(ψ (ι x . φ x)) → (∃ α. (αP) = (ι x . φ x))]]
  proof –
    have ∀ w . [(ψ (ι x . φ x)) in w] → (∃ o1 . Some o1 = dκ (ι x . φ x))
    using assms apply induct by (meta-solver; metis) +
    moreover hence
      ∀ w . [(ψ (ι x . φ x)) in w] → (that φ) = (denotation (that φ))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  $\psi$ 
  shows  $[[\psi \ x \rightarrow (\exists \ \alpha \ . \ (\alpha^P) = x)]]$ 
  proof -
    have  $\forall \ w \ . \ ([(\psi \ x) \text{ in } w] \rightarrow (\exists \ o_1 \ . \ \text{Some } o_1 = d_\kappa \ x))$ 
    using assms apply induct by (meta-solver;metis)+
    moreover hence  $\forall \ w \ . \ ([(\psi \ x) \text{ in } w] \rightarrow (x) = (\text{denotation } (x))^P)$ 
    apply transfer by (metis (mono-tags, lifting) eq-snd-iff fst-conv
option.simps(3))
    ultimately show ?thesis
    apply cut-tac unfolding identity- $\kappa$ -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]:  $[(\mathcal{A}\varphi) \equiv \varphi]$ 
  apply meta-solver by auto
lemma  $[(\mathcal{A}\varphi) \equiv \varphi]$ 
  nitpick[user-axioms, expect = genuine, card = 1, card i = 2]
  oops — Counter-model by nitpick

```

```

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

```

```

[[ $\Diamond(\exists x. (\models E!, x^P) \ \& \ \Diamond \neg(\models E!, x^P)) \ \& \ \Diamond \neg(\exists x. (\models E!, x^P) \ \& \ \Diamond \neg(\models E!, x^P))$ ]]
unfolding axiom-def
using PossiblyContingentObjectExistsAxiom
      PossiblyNoContingentObjectExistsAxiom
apply (simp add: meta-defs meta-aux conn-defs forall- $\nu$ -def
      split:  $\nu$ .split  $v$ .split)
by (metis  $\nu\nu$ - $\omega\nu$ -is- $\omega\nu$   $v$ .distinct(1)  $v$ .inject(1))

```

7.7 Axioms of Necessity and Actuality

```

lemma qml-act-1[axiom]:
  [[ $\mathcal{A}\varphi \rightarrow \Box \mathcal{A}\varphi$ ]]
by axiom-meta-solver
lemma qml-act-2[axiom]:
  [[ $\Box \varphi \equiv \mathcal{A}(\Box \varphi)$ ]]
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:
     $\exists o_1 o_2. \text{Some } o_1 = d_\kappa(x^P) \wedge \text{Some } o_2 = d_\kappa(\iota x. \varphi x) \wedge o_1 = o_2$ 
    apply cut-tac unfolding identity- $\kappa$ -def by meta-solver
  then obtain  $o_1 o_2$  where 2:
     $\text{Some } o_1 = d_\kappa(x^P) \wedge \text{Some } o_2 = d_\kappa(\iota x. \varphi x) \wedge o_1 = o_2$ 
    by auto
  hence 3:
    ( $\exists x. ((w_0 \models \varphi x) \wedge (\forall y. (w_0 \models \varphi y) \longrightarrow y = x))$ 
     $\wedge d_\kappa(\iota x. \varphi x) = \text{Some } (THE x. (w_0 \models \varphi x))$ )
    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  $\forall z. [\mathcal{A}\varphi z \text{ in } v] = [(z^P) = (x^P) \text{ in } v]$ 
proof
  fix  $z$ 
  have [ $\mathcal{A}\varphi z \text{ in } v$ ]  $\implies [(z^P) = (x^P) \text{ in } v]$ 
    unfolding identity- $\kappa$ -def apply meta-solver
    unfolding  $d_\kappa$ -def using 4 5 2 apply transfer
    apply simp by (metis  $w_0$ -def)
  moreover have [ $(z^P) = (x^P) \text{ in } v$ ]  $\implies [\mathcal{A}\varphi z \text{ in } v]$ 
    unfolding identity- $\kappa$ -def apply meta-solver
    using 2 4 5 apply transfer apply simp
    by (metis  $w_0$ -def)
  ultimately show [ $\mathcal{A}\varphi z \text{ in } v$ ] = [ $(z^P) = (x^P) \text{ in } v$ ]

```

```

      by auto
    qed
  thus  $[\forall z. \mathcal{A}\varphi z \equiv (z) = (x) \text{ in } v]$ 
    unfolding identity- $\nu$ -def
    by (simp add: AllEquivS)
next
  fix v
  assume  $[\forall z. \mathcal{A}\varphi z \equiv (z) = (x) \text{ in } v]$ 
  hence  $\bigwedge z. (dw \models \varphi z) = (\exists o_1 o_2. \text{Some } o_1 = d_\kappa(z^P) \wedge \text{Some } o_2 = d_\kappa(x^P) \wedge o_1 = o_2)$ 
  apply cut-tac unfolding identity- $\nu$ -def identity- $\kappa$ -def by meta-solver
  hence  $\forall z. \text{evalo } (\varphi z) \text{ dj } dw = (z = x)$  apply transfer by simp
  moreover hence  $\exists !x. \text{evalo } (\varphi x) \text{ dj } dw$  by metis
  ultimately have  $x^P = (\iota x. \varphi x)$  unfolding TheS by (simp add:
 $\nu\kappa$ -def)
  thus  $[x^P = (\iota x. \varphi x) \text{ in } v]$ 
    using Eq $\kappa$ S unfolding identity- $\kappa$ -def by (metis  $d_\kappa$ -proper)
  qed

```

7.9 Axioms for Complex Relation Terms

lemma *lambda-predicates-1*[*axiom*]:

$(\lambda x. \varphi x) = (\lambda y. \varphi y) ..$

lemma *lambda-predicates-2-1*[*axiom*]:

assumes *IsPropositionalInX* φ
shows $[(\lambda x. \varphi (x^P), x^P) \equiv \varphi (x^P)]$
apply *axiom-meta-solver*
using *D5-1*[*OF assms*]
apply *transfer by simp*

lemma *lambda-predicates-2-2*[*axiom*]:

assumes *IsPropositionalInXY* φ
shows $[(\lambda^2 (\lambda x y. \varphi (x^P) (y^P))), x^P, y^P] \equiv \varphi (x^P) (y^P)]$
apply *axiom-meta-solver*
using *D5-2*[*OF assms*] **apply** *transfer by simp*

lemma *lambda-predicates-2-3*[*axiom*]:

assumes *IsPropositionalInXYZ* φ
shows $[(\lambda^3 (\lambda x y z. \varphi (x^P) (y^P) (z^P))), x^P, y^P, z^P] \equiv \varphi (x^P) (y^P) (z^P)]$
proof –
have $\square[(\lambda^3 (\lambda x y z. \varphi (x^P) (y^P) (z^P))), x^P, y^P, z^P] \rightarrow \varphi (x^P) (y^P) (z^P)]$
apply *meta-solver using D5-3*[*OF assms*] **by** *auto*
moreover have
 $\square[\varphi (x^P) (y^P) (z^P) \rightarrow ((\lambda^3 (\lambda x y z. \varphi (x^P) (y^P) (z^P))), x^P, y^P, z^P)]$
apply *axiom-meta-solver*
using *D5-3*[*OF assms*] **unfolding** *d3-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*]:
 $[[(\lambda^0 \varphi) = \varphi]]$
unfolding *identity-defs*
apply *axiom-meta-solver*
by (*simp add: meta-defs meta-aux*)

lemma *lambda-predicates-3-1*[*axiom*]:
 $[[(\lambda x . \langle F, x^P \rangle) = F]]$
unfolding *identity-defs*
apply *axiom-meta-solver*
by (*simp add: meta-defs meta-aux*)

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

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

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

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

lemma *lambda-predicates-4-2*[*axiom*]:
assumes $\bigwedge x. [\mathcal{A}(\varphi x \equiv \psi x)] \text{ in } v$
shows $[(\lambda^2 (\lambda x y . \chi (\iota x. \varphi x) x y)) = (\lambda^2 (\lambda x y . \chi (\iota x. \psi x) x y))] \text{ in } v$
unfolding *identity-defs* **using** *assms* **apply** *cut-tac*
apply *meta-solver* **by** (*auto simp: meta-defs*)

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

7.10 Axioms of Encoding

lemma *encoding*[*axiom*]:
 $[[\langle x, F \rangle \rightarrow \Box \langle x, F \rangle]]$

```

    by axiom-meta-solver
lemma nocoder[axiom]:
  [[( $\langle O!, x \rangle \rightarrow \neg(\exists F . \langle x, F \rangle$ )]]
  unfolding axiom-def
  apply (rule allI, rule ImplI, subst (asm) OrdS)
  apply meta-solver unfolding en-def
  by (metis v.simps(5) mem-Collect-eq option.sel)
lemma A-objects[axiom]:
  [[( $\exists x . \langle A!, x^P \rangle \ \& \ (\forall F . (\langle x^P, F \rangle \equiv \varphi F))$ )]]
  unfolding axiom-def
  proof (rule allI, rule ExIRule)
    fix v
    let ?x =  $\alpha v \ \{ F . [\varphi F \text{ in } v] \}$ 
    have [ $\langle A!, ?x^P \rangle \text{ in } v$ ] by (simp add: AbsS d $\kappa$ -proper)
    moreover have [ $(\forall F . \langle ?x^P, F \rangle \equiv \varphi F) \text{ in } v$ ]
      apply meta-solver unfolding en-def
      using d $_1$ .rep-eq d $\kappa$ -def d $\kappa$ -proper eval $\Pi_1$ -inverse by auto
    ultimately show [ $\langle A!, ?x^P \rangle \ \& \ (\forall F . \langle ?x^P, F \rangle \equiv \varphi F) \text{ in } v$ ]
      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 propnot $_0 \equiv propnot :: \Pi_0 \Rightarrow \Pi_0$ 
      propnot $_1 \equiv propnot :: \Pi_1 \Rightarrow \Pi_1$ 
      propnot $_2 \equiv propnot :: \Pi_2 \Rightarrow \Pi_2$ 
      propnot $_3 \equiv propnot :: \Pi_3 \Rightarrow \Pi_3$ 
begin
  definition propnot $_0 :: \Pi_0 \Rightarrow \Pi_0$  where
    propnot $_0 \equiv \lambda p . \lambda^0 (\neg p)$ 
  definition propnot $_1$  where
    propnot $_1 \equiv \lambda F . \lambda x . \neg \langle F, x^P \rangle$ 
  definition propnot $_2$  where
    propnot $_2 \equiv \lambda F . \lambda^2 (\lambda x y . \neg \langle F, x^P, y^P \rangle)$ 
  definition propnot $_3$  where
    propnot $_3 \equiv \lambda F . \lambda^3 (\lambda x y z . \neg \langle F, x^P, y^P, z^P \rangle)$ 
end

named-theorems propnot-defs
declare propnot $_0$ -def[propnot-defs] propnot $_1$ -def[propnot-defs]
      propnot $_2$ -def[propnot-defs] propnot $_3$ -def[propnot-defs]

```

8.2 Noncontingent and Contingent Relations

```

consts Necessary :: 'a  $\Rightarrow$  o
overloading Necessary $_0 \equiv Necessary :: \Pi_0 \Rightarrow o$ 

```

```

    Necessary1 ≡ Necessary :: Π1⇒o
    Necessary2 ≡ Necessary :: Π2⇒o
    Necessary3 ≡ Necessary :: Π3⇒o
begin
  definition Necessary0 where
    Necessary0 ≡ λ p . □p
  definition Necessary1 :: Π1⇒o where
    Necessary1 ≡ λ F . □(∀ x . (F, xP))
  definition Necessary2 where
    Necessary2 ≡ λ F . □(∀ x y . (F, xP, yP))
  definition Necessary3 where
    Necessary3 ≡ λ F . □(∀ x y z . (F, xP, yP, zP))
end

named-theorems Necessary-defs
declare Necessary0-def[Necessary-defs] Necessary1-def[Necessary-defs]
      Necessary2-def[Necessary-defs] Necessary3-def[Necessary-defs]

consts Impossible :: 'a⇒o
overloading Impossible0 ≡ Impossible :: Π0⇒o
      Impossible1 ≡ Impossible :: Π1⇒o
      Impossible2 ≡ Impossible :: Π2⇒o
      Impossible3 ≡ Impossible :: Π3⇒o
begin
  definition Impossible0 where
    Impossible0 ≡ λ p . □¬p
  definition Impossible1 where
    Impossible1 ≡ λ F . □(∀ x . ¬(F, xP))
  definition Impossible2 where
    Impossible2 ≡ λ F . □(∀ x y . ¬(F, xP, yP))
  definition Impossible3 where
    Impossible3 ≡ λ F . □(∀ x y z . ¬(F, xP, yP, zP))
end

named-theorems Impossible-defs
declare Impossible0-def[Impossible-defs] Impossible1-def[Impossible-defs]
      Impossible2-def[Impossible-defs] Impossible3-def[Impossible-defs]

definition NonContingent where
  NonContingent ≡ λ F . (Necessary F) ∨ (Impossible F)
definition Contingent where
  Contingent ≡ λ F . ¬(Necessary F ∨ Impossible F)

definition ContingentlyTrue :: o⇒o where
  ContingentlyTrue ≡ λ p . p & ◇¬p
definition ContingentlyFalse :: o⇒o where
  ContingentlyFalse ≡ λ p . ¬p & ◇p

definition WeaklyContingent where
  WeaklyContingent ≡ λ F . Contingent F & (∀ x . ◇(F, xP) → □(F, xP))

```

8.3 Null and Universal Objects

```

definition Null :: κ⇒o where

```


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

definition $NullObject :: \kappa \ (a_\emptyset)$ **where**
 $NullObject \equiv (\iota x . Null \ (x^P))$
definition $UniversalObject :: \kappa \ (a_V)$ **where**
 $UniversalObject \equiv (\iota x . Universal \ (x^P))$

8.4 Propositional Properties

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

8.5 Indiscriminate Properties

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

8.6 Miscellaneous

definition $not_identical_E :: \kappa \Rightarrow \kappa \Rightarrow o$ (**infixl** \neq_E 63)
where $not_identical_E \equiv \lambda x \ y . \langle (\lambda^2 (\lambda x \ y . x^P =_E y^P))^- , x, y \rangle$

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
 $= ((assumption \mid (match \ axiom \ in \ A: [[\varphi]] \ for \ \varphi \Rightarrow \langle fact \ A[axiom-instance] \rangle)$
 $\mid fact \ PLM \mid rule \ PLM-intro \mid subst \ PLM-subst \mid subst \ (asm)$
 $PLM-subst$
 $\mid fastforce \mid safe \mid drule \ PLM-dest \mid erule \ PLM-elim); (PLM-solver)?$

9.2 Modus Ponens

lemma $modus-ponens[PLM]$:
 $\llbracket [\varphi \ in \ v]; [\varphi \rightarrow \psi \ in \ v] \rrbracket \Longrightarrow [\psi \ in \ v]$
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]:
   $\llbracket [\varphi \text{ in } v]; [\varphi \rightarrow \psi \text{ in } v] \rrbracket \Longrightarrow [\psi \text{ in } v]$ 
  using modus-ponens .
lemma vdash-properties-9[PLM]:
   $[\varphi \text{ in } v] \Longrightarrow [\psi \rightarrow \varphi \text{ in } v]$ 
  using modus-ponens pl-1 axiom-instance by blast
lemma vdash-properties-10[PLM]:
   $[\varphi \rightarrow \psi \text{ in } v] \Longrightarrow ([\varphi \text{ in } v] \Longrightarrow [\psi \text{ in } v])$ 
  using vdash-properties-6 .

attribute-setup deduction =  $\langle\langle$ 
  Scan.succeed (Thm.rule-attribute [])
  (fn - => fn thm => thm RS @{thm vdash-properties-10})
 $\rangle\rangle$ 
```

9.5 GEN and RN

```
lemma rule-gen[PLM]:
   $\llbracket \bigwedge \alpha . [\varphi \alpha \text{ in } v] \rrbracket \Longrightarrow [\forall \alpha . \varphi \alpha \text{ in } v]$ 
  by (simp add: Semantics.T8)

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

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

9.6 Negations and Conditionals

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

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

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

lemma useful-tautologies-1[PLM]:
```

```

[¬¬φ → φ in v]
by (meson pl-1 pl-3 ded-thm-cor-3 ded-thm-cor-4 axiom-instance)
lemma useful-tautologies-2[PLM]:
[φ → ¬¬φ 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]:
[¬φ → (φ → ψ) 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]:
[(¬ψ → ¬φ) → (φ → ψ) 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]:
[(φ → ψ) → (¬ψ → ¬φ) in v]
by (metis CP useful-tautologies-4 vdash-properties-10)
lemma useful-tautologies-6[PLM]:
[(φ → ¬ψ) → (ψ → ¬φ) in v]
by (metis CP useful-tautologies-4 vdash-properties-10)
lemma useful-tautologies-7[PLM]:
[(¬φ → ψ) → (¬ψ → φ) in v]
using ded-thm-cor-3 useful-tautologies-4 useful-tautologies-5
useful-tautologies-6 by blast
lemma useful-tautologies-8[PLM]:
[φ → (¬ψ → ¬(φ → ψ)) in v]
by (meson ded-thm-cor-3 CP useful-tautologies-5)
lemma useful-tautologies-9[PLM]:
[(φ → ψ) → ((¬φ → ψ) → ψ) in v]
by (metis CP useful-tautologies-4 vdash-properties-10)
lemma useful-tautologies-10[PLM]:
[(φ → ¬ψ) → ((φ → ψ) → ¬φ) in v]
by (metis ded-thm-cor-3 CP useful-tautologies-6)

lemma modus-tollens-1[PLM]:
[[φ → ψ in v]; [¬ψ in v]] ⇒ [¬φ 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]:
[[φ → ¬ψ in v]; [ψ in v]] ⇒ [¬φ in v]
using modus-tollens-1 useful-tautologies-2
vdash-properties-10 by blast

lemma contraposition-1[PLM]:
[φ → ψ in v] = [¬ψ → ¬φ in v]
using useful-tautologies-4 useful-tautologies-5
vdash-properties-10 by blast
lemma contraposition-2[PLM]:
[φ → ¬ψ in v] = [ψ → ¬φ in v]
using contraposition-1 ded-thm-cor-3
useful-tautologies-1 by blast

lemma reductio-aa-1[PLM]:
[[¬φ in v] ⇒ [¬ψ in v]; [¬φ in v] ⇒ [ψ in v]] ⇒ [φ in v]
using CP modus-tollens-2 useful-tautologies-1
vdash-properties-10 by blast

```

lemma *reductio-aa-2*[PLM]:

$$\llbracket [\varphi \text{ in } v] \implies [\neg\psi \text{ in } v]; [\varphi \text{ in } v] \implies [\psi \text{ in } v] \rrbracket \implies [\neg\varphi \text{ in } v]$$
by (*meson contraposition-1 reductio-aa-1*)

lemma *reductio-aa-3*[PLM]:

$$\llbracket [\neg\varphi \rightarrow \neg\psi \text{ in } v]; [\neg\varphi \rightarrow \psi \text{ in } v] \rrbracket \implies [\varphi \text{ in } v]$$
using *reductio-aa-1 vdash-properties-10* **by** *blast*

lemma *reductio-aa-4*[PLM]:

$$\llbracket [\varphi \rightarrow \neg\psi \text{ in } v]; [\varphi \rightarrow \psi \text{ in } v] \rrbracket \implies [\neg\varphi \text{ in } v]$$
using *reductio-aa-2 vdash-properties-10* **by** *blast*

lemma *raa-cor-1*[PLM]:

$$\llbracket [\varphi \text{ in } v]; [\neg\psi \text{ in } v] \implies [\neg\varphi \text{ in } v] \rrbracket \implies ([\varphi \text{ in } v] \implies [\psi \text{ in } v])$$
using *reductio-aa-1 vdash-properties-9* **by** *blast*

lemma *raa-cor-2*[PLM]:

$$\llbracket [\neg\varphi \text{ in } v]; [\neg\psi \text{ in } v] \implies [\varphi \text{ in } v] \rrbracket \implies ([\neg\varphi \text{ in } v] \implies [\psi \text{ in } v])$$
using *reductio-aa-1 vdash-properties-9* **by** *blast*

lemma *raa-cor-3*[PLM]:

$$\llbracket [\varphi \text{ in } v]; [\neg\psi \rightarrow \neg\varphi \text{ in } v] \rrbracket \implies ([\varphi \text{ in } v] \implies [\psi \text{ in } v])$$
using *raa-cor-1 vdash-properties-10* **by** *blast*

lemma *raa-cor-4*[PLM]:

$$\llbracket [\neg\varphi \text{ in } v]; [\neg\psi \rightarrow \varphi \text{ in } v] \rrbracket \implies ([\neg\varphi \text{ in } v] \implies [\psi \text{ in } v])$$
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]:

$$\llbracket [\varphi \text{ in } v]; [\psi \text{ in } v] \rrbracket \implies [\varphi \ \& \ \psi \text{ in } v]$$
unfolding *conj-def* **using** *ded-thm-cor-4 if-p-then-p modus-tollens-2*
by *blast*

lemmas $\&I = \text{intro-elim-1}$

lemma *intro-elim-2-a*[PLM]:

$$[\varphi \ \& \ \psi \text{ in } v] \implies [\varphi \text{ in } v]$$
unfolding *conj-def* **using** *CP reductio-aa-1* **by** *blast*

lemma *intro-elim-2-b*[PLM]:

$$[\varphi \ \& \ \psi \text{ in } v] \implies [\psi \text{ in } v]$$
unfolding *conj-def* **using** *pl-1 CP reductio-aa-1 axiom-instance* **by** *blast*

lemmas $\&E = \text{intro-elim-2-a intro-elim-2-b}$

lemma *intro-elim-3-a*[PLM]:

$$[\varphi \text{ in } v] \implies [\varphi \vee \psi \text{ in } v]$$
unfolding *disj-def* **using** *ded-thm-cor-4 useful-tautologies-3* **by** *blast*

lemma *intro-elim-3-b*[PLM]:

$$[\psi \text{ in } v] \implies [\varphi \vee \psi \text{ in } v]$$
by (*simp only: disj-def vdash-properties-9*)

lemmas $\vee I = \text{intro-elim-3-a intro-elim-3-b}$

lemma *intro-elim-4-a*[PLM]:

$$\llbracket [\varphi \vee \psi \text{ in } v]; [\varphi \rightarrow \chi \text{ in } v]; [\psi \rightarrow \chi \text{ in } v] \rrbracket \implies [\chi \text{ in } v]$$
unfolding *disj-def* **by** (*meson reductio-aa-2 vdash-properties-10*)

lemma *intro-elim-4-b*[PLM]:

$$\llbracket [\varphi \vee \psi \text{ in } v]; [\neg\varphi \text{ in } v] \rrbracket \implies [\psi \text{ in } v]$$
unfolding *disj-def* **using** *vdash-properties-10* **by** *blast*

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

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

private lemma *ImplI*[PLM-intro]:
 $([\varphi \text{ in } v] \Longrightarrow [\psi \text{ in } v]) \Longrightarrow [\varphi \rightarrow \psi \text{ in } v]$

```

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

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

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

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

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

private lemma NotOrD[PLM-dest]:
   $\neg[\varphi \vee \psi \text{ in } v] \Longrightarrow \neg[\varphi \text{ in } v] \wedge \neg[\psi \text{ in } v]$ 
using  $\vee I$  by blast
private lemma NotAndD[PLM-dest]:
   $\neg[\varphi \ \& \ \psi \text{ in } v] \Longrightarrow \neg[\varphi \text{ in } v] \vee \neg[\psi \text{ in } v]$ 
using  $\&I$  by blast

```

```

private lemma NotEquivD[PLM-dest]:
   $\neg[\varphi \equiv \psi \text{ in } v] \implies [\varphi \text{ in } v] \neq [\psi \text{ in } v]$ 
  by (meson NotI contraposition-1  $\equiv I$  vdash-properties-9)

private lemma BoxI[PLM-intro]:
   $(\bigwedge v . [\varphi \text{ in } v]) \implies [\Box \varphi \text{ in } v]$ 
  using RN by blast
private lemma NotBoxD[PLM-dest]:
   $\neg[\Box \varphi \text{ in } v] \implies (\exists v . \neg[\varphi \text{ in } v])$ 
  using BoxI by blast

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

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

```

by *PLM-solver*
lemma *oth-class-taut-4-b*[*PLM*]:
 $[\varphi \equiv \neg\neg\varphi \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-a*[*PLM*]:
 $[(\varphi \rightarrow \psi) \equiv \neg(\varphi \ \& \ \neg\psi) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-b*[*PLM*]:
 $[\neg(\varphi \rightarrow \psi) \equiv (\varphi \ \& \ \neg\psi) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-c*[*PLM*]:
 $[(\varphi \rightarrow \psi) \rightarrow ((\psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \chi)) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-d*[*PLM*]:
 $[(\varphi \equiv \psi) \equiv (\neg\varphi \equiv \neg\psi) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-e*[*PLM*]:
 $[(\varphi \equiv \psi) \rightarrow ((\varphi \rightarrow \chi) \equiv (\psi \rightarrow \chi)) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-f*[*PLM*]:
 $[(\varphi \equiv \psi) \rightarrow ((\chi \rightarrow \varphi) \equiv (\chi \rightarrow \psi)) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-g*[*PLM*]:
 $[(\varphi \equiv \psi) \rightarrow ((\varphi \equiv \chi) \equiv (\psi \equiv \chi)) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-h*[*PLM*]:
 $[(\varphi \equiv \psi) \rightarrow ((\chi \equiv \varphi) \equiv (\chi \equiv \psi)) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-i*[*PLM*]:
 $[(\varphi \equiv \psi) \equiv ((\varphi \ \& \ \psi) \vee (\neg\varphi \ \& \ \neg\psi)) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-j*[*PLM*]:
 $[(\neg(\varphi \equiv \psi)) \equiv ((\varphi \ \& \ \neg\psi) \vee (\neg\varphi \ \& \ \psi)) \text{ in } v]$
 by *PLM-solver*
lemma *oth-class-taut-5-k*[*PLM*]:
 $[(\varphi \rightarrow \psi) \equiv (\neg\varphi \vee \psi) \text{ in } v]$
 by *PLM-solver*

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

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

lemma *oth-class-taut-7-b*[PLM]:

$$[(\varphi \vee (\psi \ \& \ \chi)) \equiv ((\varphi \vee \psi) \ \& \ (\varphi \vee \chi)) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-8-a*[PLM]:

$$[(\varphi \ \& \ \psi) \rightarrow \chi) \rightarrow (\varphi \rightarrow (\psi \rightarrow \chi)) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-8-b*[PLM]:

$$[(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \ \& \ \psi) \rightarrow \chi) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-9-a*[PLM]:

$$[(\varphi \ \& \ \psi) \rightarrow \varphi \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-9-b*[PLM]:

$$[(\varphi \ \& \ \psi) \rightarrow \psi \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-10-a*[PLM]:

$$[\varphi \rightarrow (\psi \rightarrow (\varphi \ \& \ \psi)) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-10-b*[PLM]:

$$[(\varphi \rightarrow (\psi \rightarrow \chi)) \equiv (\psi \rightarrow (\varphi \rightarrow \chi)) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-10-c*[PLM]:

$$[(\varphi \rightarrow \psi) \rightarrow ((\varphi \rightarrow \chi) \rightarrow (\varphi \rightarrow (\psi \ \& \ \chi))) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-10-d*[PLM]:

$$[(\varphi \rightarrow \chi) \rightarrow ((\psi \rightarrow \chi) \rightarrow ((\varphi \vee \psi) \rightarrow \chi)) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-10-e*[PLM]:

$$[(\varphi \rightarrow \psi) \rightarrow ((\chi \rightarrow \Theta) \rightarrow ((\varphi \ \& \ \chi) \rightarrow (\psi \ \& \ \Theta))) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-10-f*[PLM]:

$$[(\varphi \ \& \ \psi) \equiv (\varphi \ \& \ \chi)) \equiv (\varphi \rightarrow (\psi \equiv \chi)) \text{ in } v]$$
by *PLM-solver*

lemma *oth-class-taut-10-g*[PLM]:

$$[(\varphi \ \& \ \psi) \equiv (\chi \ \& \ \psi)) \equiv (\psi \rightarrow (\varphi \equiv \chi)) \text{ in } v]$$
by *PLM-solver*

attribute-setup *equiv-lr* = \ll

$$\text{Scan.succeed } (Thm.rule_attribute \ []$$

$$(fn \ - \Rightarrow fn \ thm \Rightarrow thm \ RS \ @\{thm \equiv E(1)\}))$$

$$\gg$$

attribute-setup *equiv-rl* = \ll

$$\text{Scan.succeed } (Thm.rule_attribute \ []$$

$$(fn \ - \Rightarrow fn \ thm \Rightarrow thm \ RS \ @\{thm \equiv E(2)\}))$$

$$\gg$$

attribute-setup *equiv-sym* = \ll

$$\text{Scan.succeed } (Thm.rule_attribute \ []$$

$$(fn \ - \Rightarrow fn \ thm \Rightarrow thm \ RS \ @\{thm \ oth_class_taut_3_g[equiv_lr]\}))$$

$$\gg$$

```

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::Π1) = F in v]
  unfolding identity-defs by PLM-solver
lemma id-eq-prop-prop-2[PLM]:
  [((F::Π1) = G) → (G = F) in v]
  by (meson id-eq-prop-prop-1 CP ded-thm-cor-3 l-identity[axiom-instance])
lemma id-eq-prop-prop-3[PLM]:
  [(((F::Π1) = G) & (G = H)) → (F = H) in v]
  by (metis l-identity[axiom-instance] ded-thm-cor-4 CP &E)
lemma id-eq-prop-prop-4-a[PLM]:
  [(F::Π2) = F in v]
  unfolding identity-defs by PLM-solver
lemma id-eq-prop-prop-4-b[PLM]:
  [(F::Π3) = F in v]
  unfolding identity-defs by PLM-solver
lemma id-eq-prop-prop-5-a[PLM]:
  [((F::Π2) = G) → (G = F) in v]
  by (meson id-eq-prop-prop-4-a CP ded-thm-cor-3 l-identity[axiom-instance])
lemma id-eq-prop-prop-5-b[PLM]:
  [((F::Π3) = G) → (G = F) in v]
  by (meson id-eq-prop-prop-4-b CP ded-thm-cor-3 l-identity[axiom-instance])
lemma id-eq-prop-prop-6-a[PLM]:
  [(((F::Π2) = G) & (G = H)) → (F = H) in v]
  by (metis l-identity[axiom-instance] ded-thm-cor-4 CP &E)
lemma id-eq-prop-prop-6-b[PLM]:
  [(((F::Π3) = G) & (G = H)) → (F = H) in v]
  by (metis l-identity[axiom-instance] ded-thm-cor-4 CP &E)
lemma id-eq-prop-prop-7[PLM]:
  [(p::Π0) = p in v]
  unfolding identity-defs by PLM-solver
lemma id-eq-prop-prop-7-b[PLM]:

```

```

[[p::o] = p in v]
unfolding identity-defs by PLM-solver
lemma id-eq-prop-prop-8[PLM]:
  [[(p::Π0) = q] → (q = p) in v]
by (meson id-eq-prop-prop-7 CP ded-thm-cor-3 l-identity[axiom-instance])
lemma id-eq-prop-prop-8-b[PLM]:
  [[(p::o) = q] → (q = p) in v]
by (meson id-eq-prop-prop-7-b CP ded-thm-cor-3 l-identity[axiom-instance])
lemma id-eq-prop-prop-9[PLM]:
  [(((p::Π0) = q) & (q = r)) → (p = r) in v]
by (metis l-identity[axiom-instance] ded-thm-cor-4 CP &E)
lemma id-eq-prop-prop-9-b[PLM]:
  [(((p::o) = q) & (q = r)) → (p = r) in v]
by (metis l-identity[axiom-instance] ded-thm-cor-4 CP &E)

lemma eq-E-simple-1[PLM]:
  [(x =E y) ≡ ((O!,x) & (O!,y) & □(∀ F . (F,x) ≡ (F,y))) in v]
proof (rule ≡I; rule CP)
  assume 1: [x =E y in v]
  have [∀ x y . ((xP) =E (yP)) ≡ ((O!,xP) & (O!,yP)
    & □(∀ F . (F,xP) ≡ (F,yP))) in v]
    unfolding identityE-infix-def identityE-def
  apply (rule lambda-predicates-2-2[axiom-universal, axiom-universal,
    axiom-instance])
    by (rule IsPropositional-intros)
  moreover have [∃ α . (αP) = x in v]
  apply (rule cqt-5-mod[where ψ=λ x . x =E y, axiom-instance, deduction])
    unfolding identityE-infix-def
    apply (rule SimpleExOrEnc.intros)
    using 1 unfolding identityE-infix-def by auto
  moreover have [∃ β . (βP) = y in v]
  apply (rule cqt-5-mod[where ψ=λ y . x =E y, axiom-instance, deduction])
    unfolding identityE-infix-def
    apply (rule SimpleExOrEnc.intros) using 1
    unfolding identityE-infix-def by auto
  ultimately have [(x =E y) ≡ ((O!,x) & (O!,y)
    & □(∀ F . (F,x) ≡ (F,y))) in v]
    using cqt-1-κ[axiom-instance, deduction, deduction] by meson
  thus [(O!,x) & (O!,y) & □(∀ F . (F,x) ≡ (F,y))] in v
    using 1 ≡E(1) by blast
next
  assume 1: [(O!,x) & (O!,y) & □(∀ F . (F,x) ≡ (F,y))] in v]
  have [∀ x y . ((xP) =E (yP)) ≡ ((O!,xP) & (O!,yP)
    & □(∀ F . (F,xP) ≡ (F,yP))) in v]
    unfolding identityE-def identityE-infix-def
  apply (rule lambda-predicates-2-2[axiom-universal, axiom-universal,
    axiom-instance])
    by (rule IsPropositional-intros)
  moreover have [∃ α . (αP) = x in v]
  apply (rule cqt-5-mod[where ψ=λ x . (O!,x), axiom-instance, deduction])
    apply (rule SimpleExOrEnc.intros)
    using 1 [conj1, conj1] by auto
  moreover have [∃ β . (βP) = y in v]
  apply (rule cqt-5-mod[where ψ=λ y . (O!,y), axiom-instance, deduction])

```

```

    apply (rule SimpleExOrEnc.intros)
    using 1[conj1,conj2] by auto
    ultimately have  $[(x =_E y) \equiv (\langle O!,x \rangle \ \& \ \langle O!,y \rangle)$ 
       $\& \ \Box(\forall F. \langle F,x \rangle \equiv \langle F,y \rangle)]$  in v]
    using cqt-1- $\kappa$ [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]:
 $[(x = y) \equiv (((\langle O!,x \rangle \ \& \ \langle O!,y \rangle \ \& \ \Box(\forall F. \langle F,x \rangle \equiv \langle F,y \rangle))$ 
   $\vee (\langle A!,x \rangle \ \& \ \langle A!,y \rangle \ \& \ \Box(\forall F. \langle x,F \rangle \equiv \langle y,F \rangle)))]$  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  $[(\Diamond \langle E!, x^P \rangle) \vee (\neg \Diamond \langle E!, x^P \rangle)]$  in v]
      using PLM.oth-class-taut-2 by simp
    hence  $[(\Diamond \langle E!, x^P \rangle)$  in v]  $\vee [(\neg \Diamond \langle E!, x^P \rangle)$  in v]
      using CP  $\vee E(1)$  by blast
    moreover {
      assume  $[(\Diamond \langle E!, x^P \rangle)$  in v]
      hence  $[(\langle \lambda x. \Diamond \langle E!,x^P \rangle, x^P \rangle)$  in v]
        apply (rule lambda-predicates-2-1[axiom-instance, equiv-rl, ro-
tated])
        by (rule IsPropositional-intros)+
      hence  $[(\langle \lambda x. \Diamond \langle E!,x^P \rangle, x^P \rangle \ \& \ \langle \lambda x. \Diamond \langle E!,x^P \rangle, x^P \rangle)$ 
         $\& \ \Box(\forall F. \langle F,x^P \rangle \equiv \langle F,x^P \rangle)]$  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  $[(\neg \Diamond \langle E!, x^P \rangle)$  in v]
      hence  $[(\langle \lambda x. \neg \Diamond \langle E!,x^P \rangle, x^P \rangle)$  in v]
        apply (rule lambda-predicates-2-1[axiom-instance, equiv-rl, ro-
tated])
        by (rule IsPropositional-intros)+
      hence  $[(\langle \lambda x. \neg \Diamond \langle E!,x^P \rangle, x^P \rangle \ \& \ \langle \lambda x. \neg \Diamond \langle E!,x^P \rangle, x^P \rangle)$ 
         $\& \ \Box(\forall F. \langle x^P, F \rangle \equiv \langle x^P, F \rangle)]$  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:  $[(x :: 'a) = x \text{ in } v]$ 
  assumes id-eq-2:  $[(x :: 'a) = y \rightarrow (y = x) \text{ in } v]$ 
  assumes id-eq-3:  $[(x :: 'a) = y \ \&\ (y = z) \rightarrow (x = z) \text{ in } v]$ 

```

```

instantiation  $\nu :: id\text{-eq}$ 
begin
  instance proof
    fix  $x :: \nu$  and  $v$ 
    show  $[x = x \text{ in } v]$ 
      using PLM.id-eq-obj-1
      by (simp add: identity- $\nu$ -def)
    next
    fix  $x y :: \nu$  and  $v$ 
    show  $[x = y \rightarrow y = x \text{ in } v]$ 
      using PLM.id-eq-obj-2
      by (simp add: identity- $\nu$ -def)
    next
    fix  $x y z :: \nu$  and  $v$ 
    show  $[(x = y) \ \&\ (y = z) \rightarrow x = z \text{ in } v]$ 
      using PLM.id-eq-obj-3
      by (simp add: identity- $\nu$ -def)
  qed
end

```

```

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

```

```

instantiation  $\Pi_1 :: id\text{-eq}$ 
begin
  instance proof
    fix  $x :: \Pi_1$  and  $v$ 
    show  $[x = x \text{ in } v]$ 
      using PLM.id-eq-prop-prop-1 .
  qed
end

```

```

next
  fix x y ::  $\Pi_1$  and v
  show  $[x = y \rightarrow y = x \text{ in } v]$ 
  using PLM.id-eq-prop-prop-2 .
next
  fix x y z ::  $\Pi_1$  and v
  show  $[(x = y) \ \& \ (y = z)) \rightarrow x = z \text{ in } v]$ 
  using PLM.id-eq-prop-prop-3 .
qed
end

instantiation  $\Pi_2 :: \text{id-eq}$ 
begin
  instance proof
    fix x ::  $\Pi_2$  and v
    show  $[x = x \text{ in } v]$ 
    using PLM.id-eq-prop-prop-4-a .
  next
    fix x y ::  $\Pi_2$  and v
    show  $[x = y \rightarrow y = x \text{ in } v]$ 
    using PLM.id-eq-prop-prop-5-a .
  next
    fix x y z ::  $\Pi_2$  and v
    show  $[(x = y) \ \& \ (y = z)) \rightarrow x = z \text{ in } v]$ 
    using PLM.id-eq-prop-prop-6-a .
  qed
end

instantiation  $\Pi_3 :: \text{id-eq}$ 
begin
  instance proof
    fix x ::  $\Pi_3$  and v
    show  $[x = x \text{ in } v]$ 
    using PLM.id-eq-prop-prop-4-b .
  next
    fix x y ::  $\Pi_3$  and v
    show  $[x = y \rightarrow y = x \text{ in } v]$ 
    using PLM.id-eq-prop-prop-5-b .
  next
    fix x y z ::  $\Pi_3$  and v
    show  $[(x = y) \ \& \ (y = z)) \rightarrow x = z \text{ in } v]$ 
    using PLM.id-eq-prop-prop-6-b .
  qed
end

context PLM
begin
  lemma id-eq-1[PLM]:
     $[(x :: 'a :: \text{id-eq}) = x \text{ in } v]$ 
    using id-eq-1 .
  lemma id-eq-2[PLM]:
     $[(x :: 'a :: \text{id-eq}) = y) \rightarrow (y = x) \text{ in } v]$ 
    using id-eq-2 .
  lemma id-eq-3[PLM]:

```

```

[ $((x :: 'a :: id\text{-}eq) = y) \ \&\ (y = z) \rightarrow (x = z)$  in  $v$ ]
using  $id\text{-}eq\text{-}3$  .

```

```

attribute-setup  $eq\text{-}sym = \langle\langle$ 
   $Scan.succeed$  ( $Thm.rule\text{-}attribute$  [])
  ( $fn \text{-} => fn \textit{thm} => \textit{thm} \textit{RS} @\{\textit{thm} \textit{id}\text{-}eq\text{-}2[\textit{deduction}]\}$ )
 $\rangle\rangle$ 

```

```

lemma  $all\text{-}self\text{-}eq\text{-}1[PLM]$ :
  [ $\Box(\forall \alpha :: 'a :: id\text{-}eq . \alpha = \alpha)$  in  $v$ ]
  by  $PLM\text{-}solver$ 
lemma  $all\text{-}self\text{-}eq\text{-}2[PLM]$ :
  [ $\forall \alpha :: 'a :: id\text{-}eq . \Box(\alpha = \alpha)$  in  $v$ ]
  by  $PLM\text{-}solver$ 

```

```

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

```

```

lemma  $t\text{-}id\text{-}t\text{-}proper\text{-}2[PLM]$ : [ $\tau = \tau' \rightarrow (\exists \beta . (\beta^P) = \tau')$  in  $v$ ]
proof ( $rule \textit{CP}$ )
  assume [ $\tau = \tau'$  in  $v$ ]
  moreover {
    assume [ $\tau =_E \tau'$  in  $v$ ]
    hence [ $\exists \beta . (\beta^P) = \tau'$  in  $v$ ]
    apply  $cut\text{-}tac$ 
    apply ( $rule \textit{cqt}\text{-}5\text{-}mod[\textit{where} \ \psi = \lambda \tau' . \tau =_E \tau', \textit{axiom}\text{-}instance,$ 
 $\textit{deduction}]$ )
    subgoal unfolding  $identity\text{-}defs$  by ( $rule \textit{SimpleExOrEnc.intros}$ )
    by  $simp$ 
  }

```

```

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

lemma id-nec[PLM]:  $[\langle (\alpha::'a::id-eq) = (\beta) \rangle \equiv \Box(\langle \alpha \rangle = \langle \beta \rangle) \text{ in } v]$ 
apply (rule  $\equiv I$ )
using l-identity[where  $\varphi = (\lambda \beta. \Box(\langle \alpha \rangle = \langle \beta \rangle))$ , 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]:
 $[\langle (\iota x. \varphi x) = (\iota x. \psi x) \rangle \equiv \Box(\langle \iota x. \varphi x \rangle = \langle \iota x. \psi x \rangle) \text{ in } v]$ 
proof cases  $[\langle \exists \alpha. (\alpha^P) = (\iota x. \varphi x) \rangle \text{ in } v] \wedge [\langle \exists \beta. (\beta^P) = (\iota x. \psi x) \rangle \text{ in } v]$ 
assume  $[\langle \exists \alpha. (\alpha^P) = (\iota x. \varphi x) \rangle \text{ in } v] \wedge [\langle \exists \beta. (\beta^P) = (\iota x. \psi x) \rangle \text{ in } v]$ 
then obtain  $\alpha$  and  $\beta$  where
 $[\langle \alpha^P \rangle = (\iota x. \varphi x) \text{ in } v] \wedge [\langle \beta^P \rangle = (\iota x. \psi x) \text{ in } v]$ 
apply cut-tac unfolding conn-defs by PLM-solver
moreover {
  moreover have  $[\langle \alpha \rangle = \langle \beta \rangle \equiv \Box(\langle \alpha \rangle = \langle \beta \rangle) \text{ in } v]$  by PLM-solver
  ultimately have  $[\langle (\iota x. \varphi x) = (\beta^P) \rangle \equiv \Box(\langle \iota x. \varphi x \rangle = \langle \beta^P \rangle)] \text{ in } v]$ 
using l-identity[where  $\varphi = \lambda \alpha. \langle \alpha \rangle = \langle \beta^P \rangle \equiv \Box(\langle \alpha \rangle = \langle \beta^P \rangle)$ ,
axiom-instance]
modus-ponens unfolding identity-ν-def by metis
}
ultimately show ?thesis
using l-identity[where  $\varphi = \lambda \alpha. (\iota x. \varphi x) = \langle \alpha \rangle$ 
 $\equiv \Box(\langle \iota x. \varphi x \rangle = \langle \alpha \rangle)$ , axiom-instance]
modus-ponens by metis
next
assume  $\neg[\langle \exists \alpha. (\alpha^P) = (\iota x. \varphi x) \rangle \text{ in } v] \wedge [\langle \exists \beta. (\beta^P) = (\iota x. \psi x) \rangle \text{ in } v]$ 
hence  $\neg[\langle A!, (\iota x. \varphi x) \rangle] \text{ in } v] \wedge \neg[\langle \iota x. \varphi x \rangle =_E (\iota x. \psi x) \text{ in } v]$ 
 $\vee \neg[\langle A!, (\iota x. \psi x) \rangle] \text{ in } v] \wedge \neg[\langle \iota x. \varphi x \rangle =_E (\iota x. \psi x) \text{ in } v]$ 
unfolding identityE-infix-def
using cqt-5[axiom-instance] PLM.contraposition-1 SimpleExOrEnc.intros
vdash-properties-10 by meson
hence  $\neg[\langle \iota x. \varphi x \rangle = (\iota x. \psi x) \text{ in } v]$ 
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*]:
 $[\forall \alpha . \varphi \alpha \text{ in } v] \implies [\varphi \beta \text{ in } v]$
by (*meson cqt-1*[*axiom-instance, deduction*])
lemmas $\forall E = \text{rule-ui}$

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

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

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

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

lemma *cqt-basic-12*[*PLM*]:

$[(\forall \alpha. \varphi \alpha) \equiv (\forall \beta. \varphi \beta) \text{ in } v]$

by *PLM-solver*

lemma *existential*[*PLM*,*PLM-intro*]:

$[\varphi \alpha \text{ in } v] \implies [\exists \alpha. \varphi \alpha \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemmas $\exists I = \text{existential}$

lemma *instantiation*-[*PLM*,*PLM-elim*,*PLM-dest*]:

$[[\exists \alpha. \varphi \alpha \text{ in } v]; (\bigwedge \alpha. [\varphi \alpha \text{ in } v] \implies [\psi \text{ in } v])] \implies [\psi \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *Instantiate*:

assumes $[\exists x. \varphi x \text{ in } v]$

obtains *x* **where** $[\varphi x \text{ in } v]$

apply (*insert assms*) **unfolding** *exists-def* **by** *PLM-solver*

lemmas $\exists E = \text{Instantiate}$

lemma *cqt-further-1*[*PLM*]:

$[(\forall \alpha. \varphi \alpha) \rightarrow (\exists \alpha. \varphi \alpha) \text{ in } v]$

by *PLM-solver*

lemma *cqt-further-2*[*PLM*]:

$[(\neg(\forall \alpha. \varphi \alpha)) \equiv (\exists \alpha. \neg \varphi \alpha) \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *cqt-further-3*[*PLM*]:

$[(\forall \alpha. \varphi \alpha) \equiv \neg(\exists \alpha. \neg \varphi \alpha) \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *cqt-further-4*[*PLM*]:

$[(\neg(\exists \alpha. \varphi \alpha)) \equiv (\forall \alpha. \neg \varphi \alpha) \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *cqt-further-5*[*PLM*]:

$[(\exists \alpha. \varphi \alpha \ \& \ \psi \alpha) \rightarrow ((\exists \alpha. \varphi \alpha) \ \& \ (\exists \alpha. \psi \alpha)) \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *cqt-further-6*[*PLM*]:

$[(\exists \alpha. \varphi \alpha \vee \psi \alpha) \equiv ((\exists \alpha. \varphi \alpha) \vee (\exists \alpha. \psi \alpha)) \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *cqt-further-10*[*PLM*]:

$[(\varphi(\alpha::'a::id\text{-eq}) \ \& \ (\forall \beta. \varphi \beta \rightarrow \beta = \alpha)) \equiv (\forall \beta. \varphi \beta \equiv \beta = \alpha) \text{ in } v]$

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*]:

$[(\forall \alpha. \varphi \alpha) \ \& \ (\forall \alpha. \psi \alpha) \rightarrow (\forall \alpha. \varphi \alpha \equiv \psi \alpha) \text{ in } v]$

by *PLM-solver*

lemma *cqt-further-12*[*PLM*]:

$[(\neg(\exists \alpha. \varphi \alpha) \ \& \ (\neg(\exists \alpha. \psi \alpha))) \rightarrow (\forall \alpha. \varphi \alpha \equiv \psi \alpha) \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *cqt-further-13*[*PLM*]:

$[(\exists \alpha. \varphi \alpha \ \& \ (\neg(\exists \alpha. \psi \alpha))) \rightarrow (\neg(\forall \alpha. \varphi \alpha \equiv \psi \alpha)) \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *cqt-further-14*[*PLM*]:

$[(\exists \alpha. \exists \beta. \varphi \alpha \beta) \equiv (\exists \beta. \exists \alpha. \varphi \alpha \beta) \text{ in } v]$

unfolding *exists-def* **by** *PLM-solver*

lemma *nec-exist-unique*[*PLM*]:
 $[(\forall x. \varphi x \rightarrow \Box(\varphi x)) \rightarrow ((\exists !x. \varphi x) \rightarrow (\exists !x. \Box(\varphi x))) \text{ in } v]$
proof (*rule CP*)
assume $a: [\forall x. \varphi x \rightarrow \Box \varphi x \text{ in } v]$
show $[(\exists !x. \varphi x) \rightarrow (\exists !x. \Box \varphi x) \text{ in } v]$
proof (*rule CP*)
assume $[(\exists !x. \varphi x) \text{ in } v]$
hence $[\exists \alpha. \varphi \alpha \ \& \ (\forall \beta. \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$
 by (*simp only: exists-unique-def*)
then obtain α **where** 1:
 $[\varphi \alpha \ \& \ (\forall \beta. \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$
 by (*rule $\exists E$*)
 {
 fix β
 have $[\Box \varphi \beta \rightarrow \beta = \alpha \text{ in } v]$
 using 1 & *E*(2) *qml-2*[*axiom-instance*]
 ded-thm-cor-3 $\forall E$ **by** *fastforce*
 }
hence $[\forall \beta. \Box \varphi \beta \rightarrow \beta = \alpha \text{ in } v]$ **by** (*rule $\forall I$*)
moreover have $[\Box(\varphi \alpha) \text{ in } v]$
 using 1 & *E*(1) *a vdash-properties-10 cqt-orig-1*[*deduction*]
 by *fast*
ultimately have $[\exists \alpha. \Box(\varphi \alpha) \ \& \ (\forall \beta. \Box \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$
 using & *I $\exists I$* **by** *fast*
thus $[(\exists !x. \Box \varphi x) \text{ in } v]$
 unfolding *exists-unique-def* **by** *assumption*
qed
qed

9.9 Actuality and Descriptions

lemma *nec-imp-act*[*PLM*]: $[\Box \varphi \rightarrow \mathcal{A} \varphi \text{ in } v]$
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*]:
 $[\mathcal{A}(\mathcal{A} \varphi \rightarrow \varphi) \text{ in } v]$
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*]:
 $[\mathcal{A}(\varphi \rightarrow \mathcal{A} \varphi) \text{ in } v]$
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*]:
 $[(\mathcal{A} \varphi \ \& \ \mathcal{A} \psi) \rightarrow \mathcal{A}(\varphi \ \& \ \psi) \text{ in } v]$
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]:
  [ $\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)$  in  $v$ ]
  using logic-actual-nec-4[axiom-instance]
       $act-conj-act-4 \equiv E(1)$ 
  by blast
lemma closure-act-2[PLM]:
  [ $\forall \alpha. \mathcal{A}(\mathcal{A}(\varphi \alpha) \equiv \varphi \alpha)$  in  $v$ ]
  by PLM-solver

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

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

lemma RA-2[PLM, PLM-intro]:
  ( $[\psi$  in  $dw]$   $\implies$  [ $\varphi$  in  $dw$ ])  $\implies$  ( $[\mathcal{A}\psi$  in  $dw]$   $\implies$  [ $\mathcal{A}\varphi$  in  $dw$ ])
  using RA logic-actual intro-elim-6-a by blast

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

  private lemma NotActualD[PLM-dest]:

```

$\neg[\mathcal{A}\varphi \text{ in } dw] \implies \neg[\varphi \text{ in } dw]$
using *RA* **by** *metis*

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

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

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

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

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

private lemma *ActualDisjI*[*PLM-intro*]:
 $[\mathcal{A}\varphi \vee \mathcal{A}\psi \text{ in } v] \implies [\mathcal{A}(\varphi \vee \psi) \text{ in } v]$
unfolding *disj-def* **by** *PLM-solver*
private lemma *ActualDisjE*[*PLM-elim*, *PLM-dest*]:
 $[\mathcal{A}(\varphi \vee \psi) \text{ in } v] \implies [\mathcal{A}\varphi \vee \mathcal{A}\psi \text{ in } v]$

```

    unfolding disj-def by PLM-solver
private lemma NotActualDisjD[PLM-dest]:
   $\neg[\mathcal{A}(\varphi \vee \psi) \text{ in } v] \implies \neg[\mathcal{A}\varphi \vee \mathcal{A}\psi \text{ in } v]$ 
  using ActualDisjI by blast

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

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

lemma ANeg-1[PLM]:
   $[\neg\mathcal{A}\varphi \equiv \neg\varphi \text{ in } dw]$ 
  by PLM-solver
lemma ANeg-2[PLM]:
   $[\neg\mathcal{A}\neg\varphi \equiv \varphi \text{ in } dw]$ 
  by PLM-solver
lemma Act-Basic-1[PLM]:
   $[\mathcal{A}\varphi \vee \mathcal{A}\neg\varphi \text{ in } v]$ 
  by PLM-solver
lemma Act-Basic-2[PLM]:
   $[\mathcal{A}(\varphi \ \& \ \psi) \equiv (\mathcal{A}\varphi \ \& \ \mathcal{A}\psi) \text{ in } v]$ 
  by PLM-solver
lemma Act-Basic-3[PLM]:
   $[\mathcal{A}(\varphi \equiv \psi) \equiv ((\mathcal{A}(\varphi \rightarrow \psi)) \ \& \ (\mathcal{A}(\psi \rightarrow \varphi))) \text{ in } v]$ 
  by PLM-solver
lemma Act-Basic-4[PLM]:
   $[(\mathcal{A}(\varphi \rightarrow \psi) \ \& \ \mathcal{A}(\psi \rightarrow \varphi)) \equiv (\mathcal{A}\varphi \equiv \mathcal{A}\psi) \text{ in } v]$ 
  by PLM-solver
lemma Act-Basic-5[PLM]:
   $[\mathcal{A}(\varphi \equiv \psi) \equiv (\mathcal{A}\varphi \equiv \mathcal{A}\psi) \text{ in } v]$ 
  by PLM-solver
lemma Act-Basic-6[PLM]:
   $[\Diamond\varphi \equiv \mathcal{A}(\Diamond\varphi) \text{ in } v]$ 
  unfolding diamond-def by PLM-solver
lemma Act-Basic-7[PLM]:
   $[\mathcal{A}\varphi \equiv \Box\mathcal{A}\varphi \text{ in } v]$ 
  by (simp add: qml-2[axiom-instance] qml-act-1[axiom-instance]  $\equiv I$ )
lemma Act-Basic-8[PLM]:

```

```

[ $\mathcal{A}(\Box\varphi) \rightarrow \Box\mathcal{A}\varphi$  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]:
[ $\Box\varphi \rightarrow \Box\mathcal{A}\varphi$  in  $v$ ]
using qml-act-1[axiom-instance] ded-thm-cor-3 nec-imp-act by blast
lemma Act-Basic-10[PLM]:
[ $\mathcal{A}(\varphi \vee \psi) \equiv \mathcal{A}\varphi \vee \mathcal{A}\psi$  in  $v$ ]
by PLM-solver

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

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

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

lemma hintikka[PLM]:
[ $(x^P = (\iota x. \varphi x)) \equiv (\varphi x \ \&\ (\forall z. \varphi z \rightarrow z = x))$  in  $dw$ ]
proof -
have [ $(\forall z. \varphi z \equiv z = x) \equiv (\varphi x \ \&\ (\forall z. \varphi z \rightarrow z = x))$  in  $dw$ ]
unfolding identity- $\nu$ -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- $\nu$ -def by fastforce
thus ?thesis using  $\equiv E(5)$  fund-cont-desc by blast
qed

```

lemma *russell-axiom-a*[PLM]:
 $[(\langle F, \iota x. \varphi x \rangle) \equiv (\exists x. \varphi x \ \& \ (\forall z. \varphi z \rightarrow z = x) \ \& \ (\langle F, x^P \rangle)) \text{ in } dw]$
(is [*?lhs* \equiv *?rhs* in *dw*])
proof –
{
 assume 1: [*?lhs* in *dw*]
 hence $\exists \alpha. \alpha^P = (\iota x. \varphi x) \text{ in } dw$
 using *cqt-5*[*axiom-instance*, *deduction*] *SimpleExOrEnc.intros*
 by *blast*
 then obtain α **where** 2:
 $[\alpha^P = (\iota x. \varphi x) \text{ in } dw]$
 using $\exists E$ **by** *auto*
 hence 3: $[\varphi \alpha \ \& \ (\forall z. \varphi z \rightarrow z = \alpha) \text{ in } dw]$
 using *hintikka*[*equiv-lr*] **by** *simp*
 from 2 **have** $[(\iota x. \varphi x) = (\alpha^P) \text{ in } dw]$
 using *l-identity*[**where** $\alpha = \alpha^P$ **and** $\beta = \iota x. \varphi x$ **and** $\varphi = \lambda x. x = \alpha^P$,
 axiom-instance, *deduction*, *deduction*]
 id-eq-obj-1[**where** $x = \alpha$] **by** *auto*
 hence $[(\langle F, \alpha^P \rangle) \text{ in } dw]$
 using 1 *l-identity*[**where** $\beta = \alpha^P$ **and** $\alpha = \iota x. \varphi x$ **and** $\varphi = \lambda x. x = \alpha^P$,
 $(\langle F, x \rangle)$,
 axiom-instance, *deduction*, *deduction*] **by** *auto*
 with 3 **have** $[\varphi \alpha \ \& \ (\forall z. \varphi z \rightarrow z = \alpha) \ \& \ (\langle F, \alpha^P \rangle) \text{ in } dw]$ **by**
 (rule $\&I$)
 hence [*?rhs* in *dw*] **using** $\exists I$ [**where** $\alpha = \alpha$] **by** *simp*
}
moreover {
 assume [*?rhs* in *dw*]
 then obtain α **where** 4:
 $[\varphi \alpha \ \& \ (\forall z. \varphi z \rightarrow z = \alpha) \ \& \ (\langle F, \alpha^P \rangle) \text{ in } dw]$
 using $\exists E$ **by** *auto*
 hence $[\alpha^P = (\iota x. \varphi x) \text{ in } dw] \wedge [(\langle F, \alpha^P \rangle) \text{ in } dw]$
 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]:
 $[(\iota x. \varphi x, F) \equiv (\exists x. \varphi x \ \& \ (\forall z. \varphi z \rightarrow z = x) \ \& \ (\langle x^P, F \rangle)) \text{ in } dw]$
(is [*?lhs* \equiv *?rhs* in *dw*])
proof –
{
 assume 1: [*?lhs* in *dw*]
 hence $\exists \alpha. \alpha^P = (\iota x. \varphi x) \text{ in } dw$
 using *cqt-5*[*axiom-instance*, *deduction*] *SimpleExOrEnc.intros* **by**
blast
 then obtain α **where** 2: $[\alpha^P = (\iota x. \varphi x) \text{ in } dw]$ **by** (rule $\exists E$)

hence 3: $[(\varphi \alpha \ \& \ (\forall z . \varphi z \rightarrow z = \alpha)) \text{ in } dw]$
 using *hintikka[equiv-lr]* by *simp*
 from 2 have $[(\iota x. \varphi x) = \alpha^P \text{ in } dw]$
 using *l-identity*[where $\alpha = \alpha^P$ and $\beta = \iota x. \varphi x$ and $\varphi = \lambda x . x = \alpha^P$,
 $\text{axiom-instance, deduction, deduction}]$
 $\text{id-eq-obj-1}[\text{where } x = \alpha] \text{ by auto}$
 hence $[\llbracket \alpha^P, F \rrbracket \text{ in } dw]$
 using 1 *l-identity*[where $\beta = \alpha^P$ and $\alpha = \iota x. \varphi x$ and $\varphi = \lambda x . x = \llbracket x, F \rrbracket$,
 $\text{axiom-instance, deduction, deduction}]$ by *auto*
 with 3 have $[(\varphi \alpha \ \& \ (\forall z . \varphi z \rightarrow z = \alpha)) \ \& \ \llbracket \alpha^P, F \rrbracket \text{ in } dw]$
 using *&I* by *auto*
 hence $[?rhs \text{ in } dw]$ using $\exists I[\text{where } \alpha = \alpha]$ by (*simp add: identity-defs*)
 }
 moreover {
 assume $[?rhs \text{ in } dw]$
 then obtain α where 4:
 $[\varphi \alpha \ \& \ (\forall z . \varphi z \rightarrow z = \alpha) \ \& \ \llbracket \alpha^P, F \rrbracket \text{ in } dw]$
 using $\exists E$ by *auto*
 hence $[\alpha^P = (\iota x. \varphi x) \text{ in } dw] \wedge [\llbracket \alpha^P, F \rrbracket \text{ in } dw]$
 using *hintikka[equiv-rl]* & *E* by *blast*
 hence $[?lhs \text{ in } dw]$
 using *l-identity*[*axiom-instance, deduction, deduction*]
 by *fast*
 }
 ultimately show *?thesis* by *PLM-solver*
 qed

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

```

      hence [?rhs in dw] using  $\exists I$ [where  $\alpha=\alpha$ ] by (simp add:
identity-defs)
    }
    moreover {
      assume [?rhs in dw]
      then obtain  $\alpha$  where 4:
        [ $\varphi \alpha \ \& \ (\forall z. \varphi z \rightarrow z = \alpha) \ \& \ \psi(\alpha^P)$  in dw]
        using  $\exists E$  by auto
      hence [ $\alpha^P = (\iota x. \varphi x)$  in dw]  $\wedge$  [ $\psi(\alpha^P)$  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  $\alpha$  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  $\alpha$  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 $[\varphi (y^P) \text{ in } dw]$
 using $y\text{-in-2}[\text{deduction}]$ **unfolding** $\text{identity-}\nu\text{-def}$ **by** blast
 thus $[\varphi (\iota x. \varphi (x^P)) \text{ in } dw]$
 using $l\text{-identity}[\text{axiom-instance}, \text{deduction}, \text{deduction}]$ 1 **by** fast
qed

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

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

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

lemma $\text{equi-desc-descA-3}[PLM]$:
 assumes $\text{SimpleExOrEnc } \psi$
 shows $[\psi (\iota x. \varphi x) \rightarrow (\exists y. y^P = (\iota x. \mathcal{A}\varphi x)) \text{ in } v]$
proof $(\text{rule } CP)$
 assume $[\psi (\iota x. \varphi x) \text{ in } v]$
 hence $[\exists \alpha. \alpha^P = (\iota x. \varphi x) \text{ in } v]$
 using $\text{cqt-5}[OF \text{ assms}, \text{axiom-instance}, \text{deduction}]$ **by** auto
 then obtain α **where** $[\alpha^P = (\iota x. \varphi x) \text{ in } v]$ **by** $(\text{rule } \exists E)$
 hence $[\alpha^P = (\iota x. \mathcal{A}\varphi x) \text{ in } v]$
 using $\text{equi-desc-descA-1}[\text{equiv-lr}]$ **by** auto
 thus $[\exists y. y^P = (\iota x. \mathcal{A}\varphi x) \text{ in } v]$
 using $\exists I$ **by** fast
qed

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

moreover hence $[\alpha^P = (\iota x . \mathcal{A}\varphi x) \text{ in } v]$
 using *equi-desc-descA-1*[*equiv-lr*] by *auto*
 ultimately show $[(\iota x . \varphi x) = (\iota x . \mathcal{A}\varphi x) \text{ in } v]$
 using *l-identity*[*axiom-instance*, *deduction*, *deduction*] by *fast*
 qed

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

lemma *equiv-desc-eq*[*PLM*]:
 assumes $\bigwedge x. [\mathcal{A}(\varphi x \equiv \psi x) \text{ in } v]$
 shows $[(\forall x . ((x^P = (\iota x . \varphi x)) \equiv (x^P = (\iota x . \psi x)))) \text{ in } v]$
 proof(*rule* $\forall I$)
 fix x
 {
 assume $[x^P = (\iota x . \varphi x) \text{ in } v]$
 hence 1: $[\mathcal{A}\varphi x \ \& \ (\forall z . \mathcal{A}\varphi z \rightarrow z = x) \text{ in } v]$
 using *nec-hintikka-scheme*[*equiv-lr*] by *auto*
 hence 2: $[\mathcal{A}\varphi x \text{ in } v] \wedge [(\forall z . \mathcal{A}\varphi z \rightarrow z = x) \text{ in } v]$
 using $\&E$ by *blast*
 {
 fix z
 {
 assume $[\mathcal{A}\psi z \text{ in } v]$
 hence $[\mathcal{A}\varphi z \text{ in } v]$
 using *assms*[*where* $x=z$] apply *cut-tac* by *PLM-solver*
 moreover have $[\mathcal{A}\varphi z \rightarrow z = x \text{ in } v]$
 using 2 *cqt-1*[*axiom-instance*, *deduction*] by *auto*
 ultimately have $[z = x \text{ in } v]$
 using *vdash-properties-10* by *auto*
 }
 hence $[\mathcal{A}\psi z \rightarrow z = x \text{ in } v]$ by (*rule* *CP*)
 }
 hence $[(\forall z . \mathcal{A}\psi z \rightarrow z = x) \text{ in } v]$ by (*rule* $\forall I$)
 moreover have $[\mathcal{A}\psi x \text{ in } v]$
 using 1[*conj1*] *assms*[*where* $x=x$]
 apply *cut-tac* by *PLM-solver*
 ultimately have $[\mathcal{A}\psi x \ \& \ (\forall z . \mathcal{A}\psi z \rightarrow z = x) \text{ in } v]$
 by *PLM-solver*
 hence $[x^P = (\iota x . \psi x) \text{ in } v]$
 using *nec-hintikka-scheme*[*where* $\varphi=\psi$, *equiv-rl*] by *auto*
 }
 moreover {
 assume $[x^P = (\iota x . \psi x) \text{ in } v]$

```

hence 1: [ $\mathcal{A}\psi\ x \ \& \ (\forall z. \ \mathcal{A}\psi\ z \rightarrow z = x)$  in  $v$ ]
  using nec-hintikka-scheme[equiv-lr] by auto
hence 2: [ $\mathcal{A}\psi\ x$  in  $v$ ]  $\wedge$  [ $(\forall z. \ \mathcal{A}\psi\ z \rightarrow z = x)$  in  $v$ ]
  using  $\&E$  by blast
{
  fix  $z$ 
  {
    assume [ $\mathcal{A}\varphi\ z$  in  $v$ ]
    hence [ $\mathcal{A}\psi\ z$  in  $v$ ]
      using assms[where  $x=z$ ]
      apply cut-tac by PLM-solver
    moreover have [ $\mathcal{A}\psi\ z \rightarrow z = x$  in  $v$ ]
      using 2 cqt-1[axiom-instance, deduction] by auto
    ultimately have [ $z = x$  in  $v$ ]
      using vdash-properties-10 by auto
  }
  hence [ $\mathcal{A}\varphi\ z \rightarrow z = x$  in  $v$ ] by (rule CP)
}
hence [ $(\forall z. \ \mathcal{A}\varphi\ z \rightarrow z = x)$  in  $v$ ] by (rule  $\forall I$ )
moreover have [ $\mathcal{A}\varphi\ x$  in  $v$ ]
  using 1 [conj1] assms[where  $x=x$ ]
  apply cut-tac by PLM-solver
ultimately have [ $\mathcal{A}\varphi\ x \ \& \ (\forall z. \ \mathcal{A}\varphi\ z \rightarrow z = x)$  in  $v$ ]
  by PLM-solver
hence [ $x^P = (\iota x. \ \varphi\ x)$  in  $v$ ]
  using nec-hintikka-scheme[where  $\varphi=\varphi, equiv-rl$ ]
  by auto
}
ultimately show [ $x^P = (\iota x. \ \varphi\ x) \equiv (x^P) = (\iota x. \ \psi\ x)$  in  $v$ ]
  using  $\equiv I$  CP by auto
qed

```

```

lemma UniqueAux:
  assumes [ $(\mathcal{A}\varphi\ (\alpha::\nu) \ \& \ (\forall z. \ \mathcal{A}(\varphi\ z) \rightarrow z = \alpha))$  in  $v$ ]
  shows [ $(\forall z. \ (\mathcal{A}(\varphi\ z) \equiv (z = \alpha)))$  in  $v$ ]
  proof -
    {
      fix  $z$ 
      {
        assume [ $\mathcal{A}(\varphi\ z)$  in  $v$ ]
        hence [ $z = \alpha$  in  $v$ ]
          using assms[conj2, THEN cqt-1[where  $\alpha=z$ ,
            axiom-instance, deduction],
            deduction] by auto
      }
    }
    moreover {
      assume [ $z = \alpha$  in  $v$ ]
      hence [ $\alpha = z$  in  $v$ ]
        unfolding identity- $\nu$ -def
        using id-eq-obj-2[deduction] by fast
      hence [ $\mathcal{A}(\varphi\ z)$  in  $v$ ] using assms[conj1]
        using l-identity[axiom-instance, deduction,
          deduction] by fast
    }
  qed

```

```

ultimately have  $[(\mathcal{A}(\varphi z) \equiv (z = \alpha)) \text{ in } v]$ 
  using  $\equiv I$  CP by auto
}
thus  $[(\forall z . (\mathcal{A}(\varphi z) \equiv (z = \alpha))) \text{ in } v]$ 
by (rule  $\forall I$ )
qed

```

```

lemma nec-russell-axiom[PLM]:
  assumes SimpleExOrEnc  $\psi$ 
  shows  $[(\psi (\iota x. \varphi x)) \equiv (\exists x . (\mathcal{A} \varphi x \ \& \ (\forall z . \mathcal{A}(\varphi z) \rightarrow z = x))$ 
     $\& \psi (\alpha^P)) \text{ in } v]$ 
  (is  $[\text{?lhs} \equiv \text{?rhs} \text{ in } v]$ )
  proof -
    {
      assume 1:  $[\text{?lhs} \text{ in } v]$ 
      hence  $[\exists \alpha. (\alpha^P) = (\iota x. \varphi x) \text{ in } v]$ 
        using cqt-5[axiom-instance, deduction] assms by blast
      then obtain  $\alpha$  where 2:  $[(\alpha^P) = (\iota x. \varphi x) \text{ in } v]$  by (rule  $\exists E$ )
      hence  $[(\forall z . (\mathcal{A}(\varphi z) \equiv (z = \alpha))) \text{ in } v]$ 
        using descriptions[axiom-instance, equiv-lr] by auto
      hence 3:  $[(\mathcal{A} \varphi \alpha \ \& \ (\forall z . (\mathcal{A}(\varphi z) \rightarrow (z = \alpha)))) \text{ in } v]$ 
        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- $\nu$ -def
        using hintikka[equiv-lr] cqt-basic-2[equiv-lr, conj1]
        &I by fast
      from 2 have  $[(\iota x. \varphi x) = (\alpha^P) \text{ in } v]$ 
        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  $[\psi (\alpha^P) \text{ in } v]$ 
        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  $[(\mathcal{A} \varphi \alpha \ \& \ (\forall z . \mathcal{A}(\varphi z) \rightarrow (z = \alpha))) \ \& \ \psi (\alpha^P) \text{ in } v]$ 
        using &I by simp
      hence  $[\text{?rhs} \text{ in } v]$ 
        using  $\exists I$ [where  $\alpha=\alpha$ ]
        by (simp add: identity-defs)
    }
    moreover {
      assume  $[\text{?rhs} \text{ in } v]$ 
      then obtain  $\alpha$  where 4:
         $[(\mathcal{A} \varphi \alpha \ \& \ (\forall z . \mathcal{A}(\varphi z) \rightarrow z = \alpha)) \ \& \ \psi (\alpha^P) \text{ in } v]$ 
        using  $\exists E$  by auto
      hence  $[(\forall z . (\mathcal{A}(\varphi z) \equiv (z = \alpha))) \text{ in } v]$ 
        using UniqueAux &E(1) by auto
      hence  $[(\alpha^P) = (\iota x . \varphi x) \text{ in } v] \wedge [\psi (\alpha^P) \text{ in } v]$ 
        using descriptions[axiom-instance, equiv-rl]
        4[conj2] by blast
      hence  $[\text{?lhs} \text{ in } v]$ 
        using l-identity[axiom-instance, deduction,
          deduction]
    }
  qed

```

```

      by fast
    }
  ultimately show ?thesis by PLM-solver
qed

lemma actual-desc-1[PLM]:
   $[(\exists y . (y^P) = (\iota x. \varphi x)) \equiv (\exists ! x . \mathcal{A}(\varphi x)) \text{ in } v]$  (is [ $?lhs \equiv ?rhs$ 
in  $v$ ])
  proof -
    {
      assume [ $?lhs$  in  $v$ ]
      then obtain  $\alpha$  where
         $[(\alpha^P) = (\iota x. \varphi x)) \text{ in } v]$ 
        by (rule  $\exists E$ )
      hence  $[(\lambda A!. (\iota x. \varphi x)) \text{ in } v] \vee [(\alpha^P) =_E (\iota x. \varphi x) \text{ in } v]$ 
        apply (cut-tac) unfolding identity-defs by PLM-solver
      then obtain  $x$  where
         $[(\mathcal{A}\varphi x \ \& \ (\forall z . \mathcal{A}(\varphi z) \rightarrow z = x))) \text{ in } v]$ 
        using nec-russell-axiom[where  $\psi = \lambda x . (\lambda A!. x)$ , 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 identityE-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
         $[(\mathcal{A}\varphi x \ \& \ (\forall z . \mathcal{A}(\varphi z) \rightarrow z = x))) \text{ in } v]$ 
        unfolding exists-unique-def by (rule  $\exists E$ )
      hence  $[\forall z. \mathcal{A}\varphi z \equiv z = x \text{ in } v]$ 
        using UniqueAux by auto
      hence  $[(x^P) = (\iota x. \varphi x) \text{ in } v]$ 
        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]:
   $[(x^P) = (\iota x. \varphi) \rightarrow \mathcal{A}\varphi \text{ in } v]$ 
  using nec-hintikka-scheme[equiv-lr, conj1]
  by (rule CP)

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

lemma actual-desc-4[PLM]:
   $[(\exists y . ((y^P) = (\iota x. \varphi (x^P)))) \rightarrow \mathcal{A}(\varphi (\iota x. \varphi (x^P))) \text{ in } v]$ 
  proof (rule CP)

```

```

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

lemma unique-box-desc-1[PLM]:
 $[(\exists !x . \Box(\varphi x)) \rightarrow (\forall y . (y^P) = (\iota x . \varphi x) \rightarrow \varphi y) \text{ in } v]$ 
proof (rule CP)
  assume  $[(\exists !x . \Box(\varphi x)) \text{ in } v]$ 
  then obtain  $\alpha$  where 1:
     $[\Box \varphi \alpha \ \& \ (\forall \beta . \Box(\varphi \beta) \rightarrow \beta = \alpha) \text{ in } v]$ 
    unfolding exists-unique-def by (rule  $\exists E$ )
  {
    fix  $y$ 
    {
      assume  $[(y^P) = (\iota x . \varphi x) \text{ in } v]$ 
      hence  $[\mathcal{A}\varphi \alpha \rightarrow \alpha = y \text{ in } v]$ 
        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  $[\alpha = y \text{ in } v]$ 
        using 1[conj1] nec-imp-act vdash-properties-10 by blast
      hence  $[\varphi y \text{ in } v]$ 
        using 1[conj1] qml-2[axiom-instance, deduction]
          l-identity[axiom-instance, deduction, deduction]
        by fast
    }
    hence  $[(y^P) = (\iota x . \varphi x) \rightarrow \varphi y \text{ in } v]$ 
      by (rule CP)
  }
  thus  $[\forall y . (y^P) = (\iota x . \varphi x) \rightarrow \varphi y \text{ in } v]$ 
    by (rule  $\forall I$ )
qed

```

```

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

```

9.10 Necessity

```

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

```

```

lemma RM-1-b[PLM]:

```



```

( $\bigwedge v. [\chi \text{ in } v] \implies [\varphi \rightarrow \psi \text{ in } v] \implies ([\Box \chi \text{ in } v] \implies [\Box \varphi \rightarrow \Box \psi \text{ in } v])$ )
using RN-2 qml-1[axiom-instance] vdash-properties-10 by blast

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

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

lemma KBasic-1[PLM]:
 $[\Box \varphi \rightarrow \Box(\psi \rightarrow \varphi) \text{ in } v]$ 
by (simp only: pl-1[axiom-instance] RM-1)

lemma KBasic-2[PLM]:
 $[\Box(\neg \varphi) \rightarrow \Box(\varphi \rightarrow \psi) \text{ in } v]$ 
by (simp only: RM-1 useful-tautologies-3)

lemma KBasic-3[PLM]:
 $[\Box(\varphi \ \& \ \psi) \equiv \Box \varphi \ \& \ \Box \psi \text{ in } v]$ 
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]:
 $[\Box(\varphi \equiv \psi) \equiv (\Box(\varphi \rightarrow \psi) \ \& \ \Box(\psi \rightarrow \varphi)) \text{ in } v]$ 
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]:
 $[(\Box(\varphi \rightarrow \psi) \ \& \ \Box(\psi \rightarrow \varphi)) \rightarrow (\Box \varphi \equiv \Box \psi) \text{ in } v]$ 
by (metis qml-1[axiom-instance] CP  $\& E \equiv I$  vdash-properties-10)

lemma KBasic-6[PLM]:
 $[\Box(\varphi \equiv \psi) \rightarrow (\Box \varphi \equiv \Box \psi) \text{ in } v]$ 
using KBasic-4 KBasic-5 by (metis equiv-def ded-thm-cor-3  $\& E(1)$ )

lemma  $[(\Box \varphi \equiv \Box \psi) \rightarrow \Box(\varphi \equiv \psi) \text{ in } v]$ 
nitpick[expect=genuine, user-axioms, card = 1, card i = 2]
oops — countermodel as desired

lemma KBasic-7[PLM]:
 $[(\Box \varphi \ \& \ \Box \psi) \rightarrow \Box(\varphi \equiv \psi) \text{ in } v]$ 
proof (rule CP)
assume  $[\Box \varphi \ \& \ \Box \psi \text{ in } v]$ 
hence  $[\Box(\psi \rightarrow \varphi) \text{ in } v] \wedge [\Box(\varphi \rightarrow \psi) \text{ in } v]$ 
using  $\& E$  KBasic-1 vdash-properties-10 by blast
thus  $[\Box(\varphi \equiv \psi) \text{ in } v]$ 
using KBasic-4  $\equiv E(2)$  intro-elim-1 by blast

```

qed

lemma *KBasic-8[PLM]*:

$[\Box(\varphi \ \& \ \psi) \rightarrow \Box(\varphi \equiv \psi)]$ in v

using *KBasic-7 KBasic-3*

by (*metis equiv-def PLM.ded-thm-cor-3 &E(1)*)

lemma *KBasic-9[PLM]*:

$[\Box((\neg\varphi) \ \& \ (\neg\psi)) \rightarrow \Box(\varphi \equiv \psi)]$ in v

proof (*rule CP*)

assume $[\Box((\neg\varphi) \ \& \ (\neg\psi))]$ in v

hence $[\Box((\neg\varphi) \equiv (\neg\psi))]$ in v

using *KBasic-8 vdash-properties-10* **by** *blast*

moreover have $\bigwedge v. [((\neg\varphi) \equiv (\neg\psi)) \rightarrow (\varphi \equiv \psi)]$ in v

using *CP $\equiv E(2)$ oth-class-taut-5-d* **by** *blast*

ultimately show $[\Box(\varphi \equiv \psi)]$ in v

using *RM-1 PLM.vdash-properties-10* **by** *blast*

qed

lemma *rule-sub-lem-1-a[PLM]*:

$[\Box(\psi \equiv \chi)] \Rightarrow [(\neg\psi) \equiv (\neg\chi)]$ in v

using *qml-2[axiom-instance] $\equiv E(1)$ oth-class-taut-5-d*
vdash-properties-10

by *blast*

lemma *rule-sub-lem-1-b[PLM]*:

$[\Box(\psi \equiv \chi)] \Rightarrow [(\psi \rightarrow \Theta) \equiv (\chi \rightarrow \Theta)]$ in v

by (*metis equiv-def contraposition-1 CP &E(2) $\equiv I$*
 $\equiv E(1)$ rule-sub-lem-1-a)

lemma *rule-sub-lem-1-c[PLM]*:

$[\Box(\psi \equiv \chi)] \Rightarrow [(\Theta \rightarrow \psi) \equiv (\Theta \rightarrow \chi)]$ in v

by (*metis CP $\equiv I \equiv E(3) \equiv E(4) \neg\neg I$*
 $\neg\neg E$ rule-sub-lem-1-a)

lemma *rule-sub-lem-1-d[PLM]*:

$(\bigwedge x. [\Box(\psi \ x \equiv \chi \ x)] \text{ in } v) \Rightarrow [(\forall \alpha. \psi \ \alpha) \equiv (\forall \alpha. \chi \ \alpha)]$ in v

by (*metis equiv-def $\forall I$ CP &E $\equiv I$ raa-cor-1*
vdash-properties-10 rule-sub-lem-1-a $\forall E$)

lemma *rule-sub-lem-1-e[PLM]*:

$[\Box(\psi \equiv \chi)] \Rightarrow [\mathcal{A}\psi \equiv \mathcal{A}\chi]$ in v

using *Act-Basic-5 $\equiv E(1)$ nec-imp-act*
vdash-properties-10

by *blast*

lemma *rule-sub-lem-1-f[PLM]*:

$[\Box(\psi \equiv \chi)] \Rightarrow [\Box\psi \equiv \Box\chi]$ in v

using *KBasic-6 $\equiv I \equiv E(1)$ vdash-properties-9*
by *blast*

definition *Substable* :: $(o \Rightarrow o) \Rightarrow \text{bool}$ **where**

Substable $\equiv \lambda \varphi . \forall \psi \ \chi \ v . (\forall w . [\psi \equiv \chi \text{ in } w]) \longrightarrow [\varphi \ \psi \equiv \varphi \ \chi \text{ in } v]$

definition *Substable1* :: $((a::\text{quantifiable} \Rightarrow o) \Rightarrow o) \Rightarrow \text{bool}$ **where**

Substable1 $\equiv \lambda \varphi . \forall \psi \ \chi \ v . (\forall x \ w . [\psi \ x \equiv \chi \ x \text{ in } w]) \longrightarrow [\varphi \ \psi \equiv \varphi \ \chi \text{ in } v]$

definition *Substable2* :: $((a::\text{quantifiable} \Rightarrow b::\text{quantifiable} \Rightarrow o) \Rightarrow o) \Rightarrow \text{bool}$ **where**

Substable2 $\equiv \lambda \varphi . \forall \psi \ \chi \ v . (\forall x \ y \ w . [\psi \ x \ y \equiv \chi \ x \ y \text{ in } w])$

$\longrightarrow [\varphi \psi \equiv \varphi \chi \text{ in } v]$

definition *SubstableVar* :: ((var list \Rightarrow o) \Rightarrow o) \Rightarrow bool **where**
SubstableVar $\equiv \lambda \varphi . \forall \psi \chi v . (\forall x w . [\psi x \equiv \chi x \text{ in } w])$
 $\longrightarrow [\varphi \psi \equiv \varphi \chi \text{ in } v]$

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

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

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

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

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

$v])$
 $\longrightarrow ([\varphi \text{ bndvars } \psi \ x \equiv \varphi \text{ bndvars } \chi \ x \text{ in } v])$

named-theorems *Substable-intros*

lemma *SubstableVar-intro*[*Substable-intros*]:

SubstableAuxVar $\psi \implies \text{SubstableVar } (\lambda \varphi . \psi (\Theta \ x) \ \varphi \ x)$

unfolding *SubstableVar-def* *SubstableAuxVar-def* **by** *blast*

lemma *SubstableAux-bndvars-intro*[*Substable-intros*]:

SubstableAuxVar $(\lambda \text{ bndvars } \varphi \ x . \varphi (\Theta \text{ bndvars } x))$

unfolding *SubstableAuxVar-def* **using** *qml-2*[*axiom-instance*, *deduction*] **by** *blast*

lemma *SubstableAux-const-intro*[*Substable-intros*]:

SubstableAuxVar $(\lambda \text{ bndvars } \varphi \ x . \Theta \text{ bndvars } x)$

unfolding *SubstableAuxVar-def* **using** *oth-class-taut-4-a* **by** *blast*

lemma *SubstableAux-not-intro*[*Substable-intros*]:

SubstableAuxVar $\psi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x .$

$\neg(\psi (\Theta 1 \text{ bndvars } x) \ \varphi (\Theta 2 \text{ bndvars } x)))$

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 $\psi \implies \text{SubstableAuxVar } \chi \implies \text{SubstableAuxVar } (\lambda$
 $\text{bndvars } \varphi \ x .$

$(\psi (\Theta 1 \text{ bndvars } x) \ \varphi (\Theta 2 \text{ bndvars } x)) \rightarrow (\chi (\Theta 3 \text{ bndvars } x) \ \varphi (\Theta 4$
 $\text{bndvars } x)))$

unfolding *SubstableAuxVar-def* **by** (*metis* $\equiv I$ *CP intro-elim-6-a*
intro-elim-6-b)

lemma *SubstableAux-box-intro*[*Substable-intros*]:

SubstableAuxVar $\psi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x .$

$\Box(\psi (\Theta 1 \text{ bndvars } x) \ \varphi (\Theta 2 \text{ bndvars } x)))$

unfolding *SubstableAuxVar-def* **using** *rule-sub-lem-1-f* *RN* **by** *meson*

lemma *SubstableAux-actual-intro*[*Substable-intros*]:

SubstableAuxVar $\psi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x .$

$\mathcal{A}(\psi (\Theta 1 \text{ bndvars } x) \ \varphi (\Theta 2 \text{ bndvars } x)))$

unfolding *SubstableAuxVar-def* **using** *rule-sub-lem-1-e* *RN* **by** *meson*

lemma *SubstableAux-all-intro*[*Substable-intros*]:

SubstableAuxVar $\psi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x .$

$\forall y . (\psi (\Theta 1 \text{ bndvars } x \ y) \ \varphi (\Theta 2 \text{ bndvars } x \ y)))$

unfolding *SubstableAuxVar-def*

proof (*rule allI*) +

fix $\Psi \ \chi :: \text{var list} \Rightarrow o$ **and** $v \ x \text{ bndvars}$

assume *a1*: $\forall \Psi \ \chi \ v \ x \text{ bndvars} . (\forall x \ w . [\Psi \ x \equiv \chi \ x \text{ in } w])$
 $\longrightarrow [\psi \text{ bndvars } \Psi \ x \equiv \psi \text{ bndvars } \chi \ x \text{ in } v]$

{
assume *a2*: $(\forall x \ v . [\Psi \ x \equiv \chi \ x \text{ in } v])$

{
fix y
have $[\psi (\Theta 1 \text{ bndvars } x \ y) \ \Psi (\Theta 2 \text{ bndvars } x \ y)]$
 $\equiv [\psi (\Theta 1 \text{ bndvars } x \ y) \ \chi (\Theta 2 \text{ bndvars } x \ y) \text{ in } v]$
using *a1 a2* **by** *auto*

}
hence $[(\forall y . \psi (\Theta 1 \text{ bndvars } x \ y) \ \Psi (\Theta 2 \text{ bndvars } x \ y))$
 $\equiv (\forall y . \psi (\Theta 1 \text{ bndvars } x \ y) \ \chi (\Theta 2 \text{ bndvars } x \ y)) \text{ in } v]$
using *cqt-basic-3*[*deduction*] $\forall I$ **by** *fast*

```

}
thus ( $\forall x v . [\Psi x \equiv \chi x \text{ in } v]$ )  $\longrightarrow$ 
   $[(\forall y . \psi (\Theta 1 \text{ bndvars } x y) \Psi (\Theta 2 \text{ bndvars } x y))$ 
     $\equiv (\forall y . \psi (\Theta 1 \text{ bndvars } x y) \chi (\Theta 2 \text{ bndvars } x y)) \text{ in } v]$ 
  by auto
qed

```

lemma *Substable-intro*[*Substable-intros*]:
 $\text{SubstableVar } (\lambda \varphi . \psi \varphi) \implies \text{Substable } (\lambda \varphi . \psi (\lambda v . \varphi))$
unfolding *SubstableVar-def Substable-def* **by fast**

lemma *Substable1-intro*[*Substable-intros*]:
 $\text{SubstableVar } (\lambda \varphi . \psi (\lambda y . \varphi ((\text{qvar } y) \# \text{Nil}))) \implies \text{Substable1 } \psi$
unfolding *SubstableVar-def Substable1-def*
proof (*rule allI*) +
fix $\Psi :: 'a :: \text{quantifiable} \Rightarrow o$ **and** χv
assume $1: \forall \Psi \chi v.$
 $(\forall x w . [\Psi x \equiv \chi x \text{ in } w]) \longrightarrow [\psi (\lambda y . \Psi ((\text{qvar } y) \# \text{Nil}))$
 $\equiv \psi (\lambda y . \chi ((\text{qvar } y) \# \text{Nil})) \text{ in } v]$
 {
assume $(\forall x w . [\Psi x \equiv \chi x \text{ in } w])$
hence $[\psi (\lambda y . \Psi (\text{varq } (\text{hd } ((\text{qvar } y) \# \text{Nil}))))$
 $\equiv \psi (\lambda y . \chi (\text{varq } (\text{hd } ((\text{qvar } y) \# \text{Nil})))) \text{ in } v]$
using 1 **by fast**
hence $[\psi (\lambda y . \Psi y) \equiv \psi (\lambda y . \chi y) \text{ in } v]$
using *varq-qvar-id* [**where** $'a = 'a]$ **by fastforce**
 }
thus $(\forall x w . [\Psi x \equiv \chi x \text{ in } w]) \longrightarrow [\psi \Psi \equiv \psi \chi \text{ in } v]$
by blast
qed

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

lemma *SubstableAux-conj-intro*[*Substable-intros*]:
 $\text{SubstableAuxVar } \psi \implies \text{SubstableAuxVar } \chi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x.$
 $(\psi (\Theta 1 \text{ bndvars } x) \varphi (\Theta 2 \text{ bndvars } x)) \ \& \ (\chi (\Theta 3 \text{ bndvars } x) \varphi (\Theta 5 \text{ bndvars } x)))$

unfolding *conn-defs* **by** $((\text{rule } \text{Substable-intros})+; ((\text{assumption}+)?))+$

lemma *SubstableAux-disj-intro*[*Substable-intros*]:
 $\text{SubstableAuxVar } \psi \implies \text{SubstableAuxVar } \chi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x.$
 $(\psi (\Theta 1 \text{ bndvars } x) \varphi (\Theta 2 \text{ bndvars } x)) \vee (\chi (\Theta 3 \text{ bndvars } x) \varphi (\Theta 4 \text{ bndvars } x)))$

unfolding *conn-defs* **by** $((\text{rule } \text{Substable-intros})+; ((\text{assumption}+)?))+$

lemma *SubstableAux-equiv-intro*[*Substable-intros*]:
 $\text{SubstableAuxVar } \psi \implies \text{SubstableAuxVar } \chi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x.$
 $(\psi (\Theta 1 \text{ bndvars } x) \varphi (\Theta 2 \text{ bndvars } x)) \equiv (\chi (\Theta 3 \text{ bndvars } x) \varphi (\Theta 4 \text{ bndvars } x)))$

unfolding *conn-defs* **by** $((\text{rule } \text{Substable-intros})+; ((\text{assumption}+)?))+$

lemma *SubstableAux-diamond-intro*[*Substable-intros*]:
 $\text{SubstableAuxVar } \psi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x.$
 $\Diamond(\psi (\Theta 1 \text{ bndvars } x) \varphi (\Theta 2 \text{ bndvars } x)))$

unfolding *conn-defs* **by** $((\text{rule } \text{Substable-intros})+; ((\text{assumption}+)?))+$

lemma *SubstableAux-exists-intro*[*Substable-intros*]:
 $\text{SubstableAuxVar } \psi \implies \text{SubstableAuxVar } (\lambda \text{ bndvars } \varphi \ x.$
 $\exists y. (\psi (\Theta 1 \text{ bndvars } x \ y) \varphi (\Theta 2 \text{ bndvars } x \ y)))$

unfolding *conn-defs* **by** $((\text{rule } \text{Substable-intros})+; ((\text{assumption}+)?))+$

method *PLM-subst-method* **for** $\psi::o$ **and** $\chi::o =$
 $(\text{match } \text{conclusion} \text{ in } \Theta [\varphi \ \chi \text{ in } v] \text{ for } \Theta \text{ and } \varphi \text{ and } v \Rightarrow$
 $\langle (\text{rule } \text{rule-sub-nec}[\text{where } \Theta=\Theta \text{ and } \chi=\chi \text{ and } \psi=\psi \text{ and } \varphi=\varphi \text{ and } v=v],$
 $((\text{rule } \text{Substable-intros}, ((\text{assumption}+)?)); \text{fail})) \rangle)$

method *PLM-subst-goal-method* **for** $\varphi::o \Rightarrow o$ **and** $\psi::o =$
 $(\text{match } \text{conclusion} \text{ in } \Theta [\varphi \ \chi \text{ in } v] \text{ for } \Theta \text{ and } \chi \text{ and } v \Rightarrow$
 $\langle (\text{rule } \text{rule-sub-nec}[\text{where } \Theta=\Theta \text{ and } \chi=\chi \text{ and } \psi=\psi \text{ and } \varphi=\varphi \text{ and } v=v],$
 $((\text{rule } \text{Substable-intros}, ((\text{assumption}+)?)); \text{fail})) \rangle)$

method *PLM-subst1-method* **for** $\psi::('a::\text{quantifiable}) \Rightarrow o$ **and** $\chi::('a) \Rightarrow o$
 $=$
 $(\text{match } \text{conclusion} \text{ in } \Theta [\varphi \ \chi \text{ in } v] \text{ for } \Theta \text{ and } \varphi \text{ and } v \Rightarrow$
 $\langle (\text{rule } \text{rule-sub-nec1}[\text{where } \Theta=\Theta \text{ and } \chi=\chi \text{ and } \psi=\psi \text{ and } \varphi=\varphi \text{ and } v=v],$
 $((\text{rule } \text{Substable-intros}, ((\text{assumption}+)?)); \text{fail})) \rangle)$

method *PLM-subst1-goal-method* **for** $\varphi::('a::\text{quantifiable}) \Rightarrow o$ **and**
 $\psi::'a \Rightarrow o =$
 $(\text{match } \text{conclusion} \text{ in } \Theta [\varphi \ \chi \text{ in } v] \text{ for } \Theta \text{ and } \chi \text{ and } v \Rightarrow$
 $\langle (\text{rule } \text{rule-sub-nec1}[\text{where } \Theta=\Theta \text{ and } \chi=\chi \text{ and } \psi=\psi \text{ and } \varphi=\varphi \text{ and } v=v],$
 $((\text{rule } \text{Substable-intros}, ((\text{assumption}+)?)); \text{fail})) \rangle)$

method *PLM-subst2-method* **for** $\psi::'a::\text{quantifiable} \Rightarrow 'a \Rightarrow o$ **and** $\chi::'a \Rightarrow 'a \Rightarrow o$
 $=$
 $(\text{match } \text{conclusion} \text{ in } \Theta [\varphi \ \chi \text{ in } v] \text{ for } \Theta \text{ and } \varphi \text{ and } v \Rightarrow$
 $\langle (\text{rule } \text{rule-sub-nec2}[\text{where } \Theta=\Theta \text{ and } \chi=\chi \text{ and } \psi=\psi \text{ and } \varphi=\varphi \text{ and } v=v],$

$v=v]$,
 $((rule\ Substable-intros, ((assumption)+)?+; fail)))$
method *PLM-subst2-goal-method* **for** $\varphi::('a::quantifiable \Rightarrow 'a \Rightarrow o) \Rightarrow o$
and $\psi::'a \Rightarrow 'a \Rightarrow o =$
 $(match\ conclusion\ in\ \Theta\ [\varphi\ \chi\ in\ v]\ for\ \Theta\ and\ \chi\ and\ v \Rightarrow$
 $\langle (rule\ rule-sub-nec2[where\ \Theta=\Theta\ and\ \chi=\chi\ and\ \psi=\psi\ and\ \varphi=\varphi\ and$
 $v=v],$
 $((rule\ Substable-intros, ((assumption)+)?+; fail)))$

method *PLM-autosubst* =
 $(match\ premises\ in\ \bigwedge v . [\psi \equiv \chi\ in\ v]\ for\ \psi\ and\ \chi \Rightarrow$
 $\langle match\ conclusion\ in\ \Theta\ [\varphi\ \chi\ in\ v]\ for\ \Theta\ and\ \varphi\ and\ v \Rightarrow$
 $\langle (rule\ rule-sub-nec[where\ \Theta=\Theta\ and\ \chi=\chi\ and\ \psi=\psi\ and\ \varphi=\varphi\ and$
 $v=v],$
 $((rule\ Substable-intros, ((assumption)+)?+; fail))) \rangle \rangle$

method *PLM-autosubst-with uses WITH* =
 $(match\ WITH\ in\ Y: \bigwedge v . [\psi \equiv \chi\ in\ v]\ for\ \psi\ and\ \chi \Rightarrow$
 $\langle match\ conclusion\ in\ \Theta\ [\varphi\ \chi\ in\ v]\ for\ \Theta\ and\ \varphi\ and\ v \Rightarrow$
 $\langle (rule\ rule-sub-nec[where\ \Theta=\Theta\ and\ \chi=\chi\ and\ \psi=\psi\ and\ \varphi=\varphi\ and$
 $v=v],$
 $((rule\ Substable-intros)+; fail)), ((fact\ WITH)? \rangle \rangle$

method *PLM-autosubst1* =
 $(match\ premises\ in\ \bigwedge v\ x :: 'a::quantifiable . [\psi\ x \equiv \chi\ x\ in\ v]\ for\ \psi$
and $\chi \Rightarrow$
 $\langle match\ conclusion\ in\ \Theta\ [\varphi\ \chi\ in\ v]\ for\ \Theta\ and\ \varphi\ and\ v \Rightarrow$
 $\langle (rule\ rule-sub-nec1[where\ \Theta=\Theta\ and\ \chi=\chi\ and\ \psi=\psi\ and\ \varphi=\varphi\ and$
 $v=v],$
 $((rule\ Substable-intros, ((assumption)+)?+; fail))) \rangle \rangle$

method *PLM-autosubst2* =
 $(match\ premises\ in\ \bigwedge v\ (x :: 'a::quantifiable)\ (y::'a) . [\psi\ x\ y \equiv \chi\ x$
 $y\ in\ v]$
for $\psi\ and\ \chi \Rightarrow$
 $\langle match\ conclusion\ in\ \Theta\ [\varphi\ \chi\ in\ v]\ for\ \Theta\ and\ \varphi\ and\ v \Rightarrow$
 $\langle (rule\ rule-sub-nec2[where\ \Theta=\Theta\ and\ \chi=\chi\ and\ \psi=\psi\ and\ \varphi=\varphi\ and$
 $v=v],$
 $((rule\ Substable-intros, ((assumption)+)?+; fail))) \rangle \rangle$

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

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

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

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

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

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

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

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

lemma *rule-sub-remark-9*:
assumes $\bigwedge v. [(P, a) \equiv ((P, a) \ \& \ ((Q, b) \vee (\neg(Q, b))))] \text{ in } v$
and $[(P, a) = (P, a) \text{ in } v]$
shows $[(P, a) = ((P, a) \ \& \ ((Q, b) \vee (\neg(Q, b))))] \text{ in } v$
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]*:
 $[\Box\varphi \equiv \Box(\neg(\neg\varphi)) \text{ in } v]$
apply (*PLM-subst-method* $\varphi (\neg(\neg\varphi))$)
by *PLM-solver+*

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

lemma *KBasic2-3[PLM]*:
 $[\Box\varphi \equiv (\neg(\Diamond(\neg\varphi))) \text{ in } v]$

unfolding *diamond-def*
apply (*PLM-subst-method* $\varphi \neg(\neg\varphi)$)
apply *PLM-solver*
by (*simp add: oth-class-taut-4-b*)
lemmas $Df\Box = KBasic2-3$

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

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

lemma *KBasic2-6[PLM]*:
 $[\Diamond(\varphi \vee \psi) \equiv (\Diamond\varphi \vee \Diamond\psi) \text{ in } v]$
proof –
have $[\Box((\neg\varphi) \ \& \ (\neg\psi)) \equiv (\Box(\neg\varphi) \ \& \ \Box(\neg\psi)) \text{ in } v]$
using *KBasic-3* **by** *blast*
hence $[(\neg(\Diamond(\neg(\neg\varphi) \ \& \ (\neg\psi)))) \equiv (\Box(\neg\varphi) \ \& \ \Box(\neg\psi)) \text{ in } v]$
using $Df\Box$ **by** (*rule* $\equiv E(6)$)
hence $[(\neg(\Diamond(\neg(\neg\varphi) \ \& \ (\neg\psi)))) \equiv ((\neg(\Diamond\varphi)) \ \& \ (\neg(\Diamond\psi))) \text{ in } v]$
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 $[(\neg(\Diamond(\varphi \vee \psi))) \equiv ((\neg(\Diamond\varphi)) \ \& \ (\neg(\Diamond\psi))) \text{ in } v]$
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 $[(\neg(\neg(\Diamond(\varphi \vee \psi)))) \equiv (\neg((\neg(\Diamond\varphi)) \ \& \ (\neg(\Diamond\psi)))) \text{ in } v]$
by (*rule* *oth-class-taut-5-d[equiv-lr]*)
hence $[\Diamond(\varphi \vee \psi) \equiv (\neg((\neg(\Diamond\varphi)) \ \& \ (\neg(\Diamond\psi)))) \text{ in } v]$
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]*:
 $[(\Box\varphi \vee \Box\psi) \rightarrow \Box(\varphi \vee \psi) \text{ in } v]$
proof –
have $\bigwedge v . [\varphi \rightarrow (\varphi \vee \psi) \text{ in } v]$
by (*metis contraposition-1 contraposition-2 useful-tautologies-3*
disj-def)
hence $[\Box\varphi \rightarrow \Box(\varphi \vee \psi) \text{ in } v]$ **using** *RM-1* **by** *auto*
moreover {
have $\bigwedge v . [\psi \rightarrow (\varphi \vee \psi) \text{ in } v]$
by (*simp only: pl-1[axiom-instance] disj-def*)
 }

```

    hence  $[\Box\psi \rightarrow \Box(\varphi \vee \psi) \text{ in } v]$ 
    using RM-1 by auto
  }
  ultimately show ?thesis
  using oth-class-taut-10-d vdash-properties-10 by blast
qed

```

```

lemma KBasic2-8[PLM]:
 $[\Diamond(\varphi \ \& \ \psi) \rightarrow (\Diamond\varphi \ \& \ \Diamond\psi) \text{ in } v]$ 
by (metis CP RM-2 &I oth-class-taut-9-a
    oth-class-taut-9-b vdash-properties-10)

```

```

lemma KBasic2-9[PLM]:
 $[\Diamond(\varphi \rightarrow \psi) \equiv (\Box\varphi \rightarrow \Diamond\psi) \text{ in } v]$ 
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  $(\neg\varphi) \vee \psi \varphi \rightarrow \psi$ )
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]:
 $[\Diamond(\Box\varphi) \equiv (\neg(\Box\Diamond(\neg\varphi))) \text{ in } v]$ 
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]:
 $[\Diamond\Diamond\varphi \equiv (\neg(\Box\Box(\neg\varphi))) \text{ in } v]$ 
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]:  $[\Box(\varphi \vee \psi) \rightarrow (\Box\varphi \vee \Diamond\psi) \text{ in } v]$ 
proof -
  have  $[\Box(\psi \vee \varphi) \rightarrow (\Box(\neg\psi) \rightarrow \Box\varphi) \text{ in } v]$ 
  using CP RM-1-b  $\vee E(2)$  by blast
  hence  $[\Box(\psi \vee \varphi) \rightarrow (\Diamond\psi \vee \Box\varphi) \text{ in } v]$ 
  unfolding diamond-def disj-def
  by (meson CP  $\neg\neg E$  vdash-properties-6)
  thus ?thesis apply cut-tac
  apply (PLM-subst-method  $(\Diamond\psi \vee \Box\varphi) (\Box\varphi \vee \Diamond\psi)$ )
  apply (simp add: PLM.oth-class-taut-3-e)
  apply (PLM-subst-method  $(\psi \vee \varphi) (\varphi \vee \psi)$ )
  apply (simp add: PLM.oth-class-taut-3-e)
  by assumption
qed

```

```

lemma TBasic[PLM]:
 $[\varphi \rightarrow \Diamond\varphi \text{ in } v]$ 
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*]:
 $[\Diamond\Box\varphi \rightarrow \Box\varphi \text{ in } v]$
proof (*rule CP*)
assume $[\Diamond\Box\varphi \text{ in } v]$
hence $[\neg\Box\Diamond\neg\varphi \text{ in } v]$
using *KBasic2-10*[*equiv-lr*] **by** *simp*
moreover have $[\Diamond(\neg\varphi) \rightarrow \Box\Diamond(\neg\varphi) \text{ in } v]$
by (*simp add: qml-3*[*axiom-instance*])
ultimately have $[\neg\Diamond\neg\varphi \text{ in } v]$
by (*simp add: PLM.modus-tollens-1*)
thus $[\Box\varphi \text{ in } v]$
unfolding *diamond-def* **apply** *cut-tac*
apply (*PLM-subst-method* $\neg\neg\varphi \varphi$)
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*]:
 $[\Box\varphi \equiv \Diamond\Box\varphi \text{ in } v]$
using $5\Diamond \ T\Diamond \equiv I$ **by** *blast*

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

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

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

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

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

lemma *S5Basic-8*[*PLM*]:
 $[\Diamond\Diamond\varphi \rightarrow \Diamond\varphi \text{ in } v]$

using *S5Basic-6*[**where** $\varphi = \neg\varphi$, *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*]:

$[\Diamond\Diamond\varphi \equiv \Diamond\varphi \text{ in } v]$
using $4\Diamond$ *T* **by** (*rule* $\equiv I$)

lemma *S5Basic-10*[*PLM*]:

$[\Box(\varphi \vee \Box\psi) \equiv (\Box\varphi \vee \Box\psi) \text{ in } v]$
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*]:

$[\Box(\varphi \vee \Diamond\psi) \equiv (\Box\varphi \vee \Diamond\psi) \text{ in } v]$
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*]:

$[\Diamond(\varphi \ \& \ \Diamond\psi) \equiv (\Diamond\varphi \ \& \ \Diamond\psi) \text{ in } v]$
proof –
have $[\Box((\neg\varphi) \vee \Box(\neg\psi)) \equiv (\Box(\neg\varphi) \vee \Box(\neg\psi)) \text{ in } v]$
using *S5Basic-10* **by** *auto*
hence 1: $[(\neg\Box((\neg\varphi) \vee \Box(\neg\psi))) \equiv \neg(\Box(\neg\varphi) \vee \Box(\neg\psi)) \text{ in } v]$
using *oth-class-taut-5-d*[*equiv-lr*] **by** *auto*
have 2: $[(\Diamond(\neg((\neg\varphi) \vee (\neg(\Diamond\psi)))) \equiv (\neg((\neg(\Diamond\varphi)) \vee (\neg(\Diamond\psi)))) \text{ in } v]$
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* $(\neg\Box((\neg\varphi) \vee \Box(\neg\psi))) (\Diamond(\neg((\neg\varphi) \vee (\neg(\Diamond\psi))))$)
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]:
   $[\Diamond(\varphi \ \& \ (\Box\psi)) \equiv (\Diamond\varphi \ \& \ (\Box\psi)) \text{ in } v]$ 
  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]:
   $[\Box(\varphi \rightarrow (\Box\psi)) \equiv \Box(\Diamond\varphi \rightarrow \psi) \text{ in } v]$ 
  proof (rule  $\equiv I$ ; rule CP)
    assume  $[\Box(\varphi \rightarrow \Box\psi) \text{ in } v]$ 
    moreover {
      have  $\bigwedge v. [\Box(\varphi \rightarrow \Box\psi) \rightarrow (\Diamond\varphi \rightarrow \psi) \text{ in } v]$ 
      proof (rule CP)
        fix v
        assume  $[\Box(\varphi \rightarrow \Box\psi) \text{ in } v]$ 
        hence  $[\Diamond\varphi \rightarrow \Diamond\Box\psi \text{ in } v]$ 
        using K $\Diamond$ [deduction] by auto
        thus  $[\Diamond\varphi \rightarrow \psi \text{ in } v]$ 
        using B $\Diamond$  ded-thm-cor-3 by blast
      qed
      hence  $[\Box(\Box(\varphi \rightarrow \Box\psi) \rightarrow (\Diamond\varphi \rightarrow \psi)) \text{ in } v]$ 
      by (rule RN)
      hence  $[\Box(\Box(\varphi \rightarrow \Box\psi)) \rightarrow \Box((\Diamond\varphi \rightarrow \psi)) \text{ in } v]$ 
      using qml-1[axiom-instance, deduction] by auto
    }
    ultimately show  $[\Box(\Diamond\varphi \rightarrow \psi) \text{ in } v]$ 
    using S5Basic-6 CP vdash-properties-10 by meson
  next
  assume  $[\Box(\Diamond\varphi \rightarrow \psi) \text{ in } v]$ 
  moreover {
    fix v
    {
      assume  $[\Box(\Diamond\varphi \rightarrow \psi) \text{ in } v]$ 
      hence 1:  $[\Box\Diamond\varphi \rightarrow \Box\psi \text{ in } v]$ 
      using qml-1[axiom-instance, deduction] by auto
      assume  $[\varphi \text{ in } v]$ 
      hence  $[\Box\Diamond\varphi \text{ in } v]$ 
      using S5Basic-4[deduction] by auto
      hence  $[\Box\psi \text{ in } v]$ 
      using 1[deduction] by auto
    }
    hence  $[\Box(\Diamond\varphi \rightarrow \psi) \text{ in } v] \implies [\varphi \rightarrow \Box\psi \text{ in } v]$ 
    using CP by auto
  }
  ultimately show  $[\Box(\varphi \rightarrow \Box\psi) \text{ in } v]$ 
  using S5Basic-6 RN-2 vdash-properties-10 by blast
qed

```

```

lemma sc-eq-box-box-1[PLM]:
   $[\Box(\varphi \rightarrow \Box\varphi) \rightarrow (\Diamond\varphi \equiv \Box\varphi) \text{ in } v]$ 
  proof(rule CP)
    assume 1:  $[\Box(\varphi \rightarrow \Box\varphi) \text{ in } v]$ 
    hence  $[\Box(\Diamond\varphi \rightarrow \varphi) \text{ in } v]$ 

```

```

    using S5Basic-14[equiv-lr] by auto
  hence  $[\Diamond\varphi \rightarrow \varphi \text{ in } v]$ 
    using qml-2[axiom-instance, deduction] by auto
  moreover from 1 have  $[\varphi \rightarrow \Box\varphi \text{ in } v]$ 
    using qml-2[axiom-instance, deduction] by auto
  ultimately have  $[\Diamond\varphi \rightarrow \Box\varphi \text{ in } v]$ 
    using ded-thm-cor-3 by auto
  moreover have  $[\Box\varphi \rightarrow \Diamond\varphi \text{ in } v]$ 
    using qml-2[axiom-instance] T $\Diamond$ 
    by (rule ded-thm-cor-3)
  ultimately show  $[\Diamond\varphi \equiv \Box\varphi \text{ in } v]$ 
    by (rule  $\equiv I$ )
qed

```

lemma *sc-eg-box-box-2[PLM]*:

```

 $[(\Box(\varphi \rightarrow \Box\varphi) \rightarrow ((\neg\Box\varphi) \equiv (\Box(\neg\varphi))) \text{ in } v]$ 
proof (rule CP)
  assume  $[\Box(\varphi \rightarrow \Box\varphi) \text{ in } v]$ 
  hence  $[(\neg\Box(\neg\varphi)) \equiv \Box\varphi \text{ in } v]$ 
    using sc-eg-box-box-1[deduction] unfolding diamond-def by auto
  thus  $[(\neg\Box\varphi) \equiv (\Box(\neg\varphi))] \text{ in } v$ 
    by (meson CP  $\equiv I \equiv E(3)$ 
         $\equiv E(4) \neg I \neg E$ )
qed

```

lemma *sc-eg-box-box-3[PLM]*:

```

 $[(\Box(\varphi \rightarrow \Box\varphi) \ \& \ \Box(\psi \rightarrow \Box\psi)) \rightarrow ((\Box\varphi \equiv \Box\psi) \rightarrow \Box(\varphi \equiv \psi)) \text{ in } v]$ 
proof (rule CP)
  assume 1:  $[(\Box(\varphi \rightarrow \Box\varphi) \ \& \ \Box(\psi \rightarrow \Box\psi)) \text{ in } v]$ 
  {
    assume  $[\Box\varphi \equiv \Box\psi \text{ in } v]$ 
    hence  $[(\Box\varphi \ \& \ \Box\psi) \vee ((\neg\Box\varphi) \ \& \ (\neg\Box\psi))] \text{ in } v$ 
      using oth-class-taut-5-i[equiv-lr] by auto
    moreover {
      assume  $[\Box\varphi \ \& \ \Box\psi \text{ in } v]$ 
      hence  $[\Box(\varphi \equiv \psi) \text{ in } v]$ 
        using KBasic-7[deduction] by auto
    }
    moreover {
      assume  $[(\neg\Box\varphi) \ \& \ (\neg\Box\psi)] \text{ in } v$ 
      hence  $[\Box(\neg\varphi) \ \& \ \Box(\neg\psi) \text{ in } v]$ 
        using 1 &E &I sc-eg-box-box-2[deduction, equiv-lr]
        by metis
      hence  $[\Box((\neg\varphi) \ \& \ (\neg\psi)) \text{ in } v]$ 
        using KBasic-3[equiv-rl] by auto
      hence  $[\Box(\varphi \equiv \psi) \text{ in } v]$ 
        using KBasic-9[deduction] by auto
    }
  }
  ultimately have  $[\Box(\varphi \equiv \psi) \text{ in } v]$ 
    using CP  $\vee E(1)$  by blast
}
thus  $[\Box\varphi \equiv \Box\psi \rightarrow \Box(\varphi \equiv \psi) \text{ in } v]$ 
  using CP by auto
qed

```

lemma *derived-S5-rules-1-a*[PLM]:
assumes $\bigwedge v. [\chi \text{ in } v] \implies [\Diamond \varphi \rightarrow \psi \text{ in } v]$
shows $[\Box \chi \text{ in } v] \implies [\varphi \rightarrow \Box \psi \text{ in } v]$
proof –
 have $[\Box \chi \text{ in } v] \implies [\Box \Diamond \varphi \rightarrow \Box \psi \text{ in } v]$
 using *assms RM-1-b* **by** *metis*
 thus $[\Box \chi \text{ in } v] \implies [\varphi \rightarrow \Box \psi \text{ in } v]$
 using *S5Basic-4 vdash-properties-10 CP* **by** *metis*
qed

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

lemma *derived-S5-rules-2-a*[PLM]:
assumes $\bigwedge v. [\chi \text{ in } v] \implies [\varphi \rightarrow \Box \psi \text{ in } v]$
shows $[\Box \chi \text{ in } v] \implies [\Diamond \varphi \rightarrow \psi \text{ in } v]$
proof –
 have $[\Box \chi \text{ in } v] \implies [\Diamond \varphi \rightarrow \Diamond \Box \psi \text{ in } v]$
 using *RM-2-b assms* **by** *metis*
 thus $[\Box \chi \text{ in } v] \implies [\Diamond \varphi \rightarrow \psi \text{ in } v]$
 using *B \Diamond vdash-properties-10 CP* **by** *metis*
qed

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

lemma *BFs-1*[PLM]: $[(\forall \alpha. \Box(\varphi \alpha)) \rightarrow \Box(\forall \alpha. \varphi \alpha) \text{ in } v]$
proof (*rule derived-S5-rules-1-b*)
 fix v
 {
 fix α
 have $\bigwedge v. [(\forall \alpha. \Box(\varphi \alpha)) \rightarrow \Box(\varphi \alpha) \text{ in } v]$
 using *cqt-orig-1* **by** *metis*
 hence $[\Diamond(\forall \alpha. \Box(\varphi \alpha)) \rightarrow \Diamond \Box(\varphi \alpha) \text{ in } v]$
 using *RM-2* **by** *metis*
 moreover have $[\Diamond \Box(\varphi \alpha) \rightarrow (\varphi \alpha) \text{ in } v]$
 using *B \Diamond* **by** *auto*
 ultimately have $[\Diamond(\forall \alpha. \Box(\varphi \alpha)) \rightarrow (\varphi \alpha) \text{ in } v]$
 using *ded-thm-cor-3* **by** *auto*
 }
 hence $[\forall \alpha. \Diamond(\forall \alpha. \Box(\varphi \alpha)) \rightarrow (\varphi \alpha) \text{ in } v]$
 using $\forall I$ **by** *metis*
 thus $[\Diamond(\forall \alpha. \Box(\varphi \alpha)) \rightarrow (\forall \alpha. \varphi \alpha) \text{ in } v]$
 using *cqt-orig-2[deduction]* **by** *auto*
qed
lemmas *BF* = *BFs-1*

lemma *BFs-2*[PLM]:
 $[\Box(\forall \alpha. \varphi \alpha) \rightarrow (\forall \alpha. \Box(\varphi \alpha)) \text{ in } v]$

```

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

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

lemma BFs-4[PLM]:
 $[(\exists \alpha. \Diamond(\varphi \alpha)) \rightarrow \Diamond(\exists \alpha. \varphi \alpha) \text{ in } v]$ 
proof –
  have 1:  $[\Box(\forall \alpha. \neg(\varphi \alpha)) \rightarrow (\forall \alpha. \Box(\neg(\varphi \alpha))) \text{ in } v]$ 
    using CBF by auto
  have 2:  $[(\exists \alpha. (\neg(\Box(\neg(\varphi \alpha)))) \rightarrow (\neg(\Box(\forall \alpha. \neg(\varphi \alpha)))) \text{ in } v]$ 
    apply (PLM-subst-method  $\neg(\forall \alpha. \Box(\neg(\varphi \alpha))) (\exists \alpha. (\neg(\Box(\neg(\varphi \alpha))))$ )
    using cqt-further-2 apply assumption
    using 1 using contraposition-1 by metis
  have  $[(\exists \alpha. (\neg(\Box(\neg(\varphi \alpha)))) \rightarrow \Diamond(\neg(\forall \alpha. \neg(\varphi \alpha))) \text{ in } v]$ 
    apply (PLM-subst-method  $\neg(\Box(\forall \alpha. \neg(\varphi \alpha))) \Diamond(\neg(\forall \alpha. \neg(\varphi \alpha)))$ )
    using KBasic2-2 apply assumption
    using 2 by assumption
  thus ?thesis
    unfolding diamond-def exists-def by auto
qed
lemmas  $CBF\Diamond = BFs-4$ 

lemma sign-S5-thm-1[PLM]:
 $[(\exists \alpha. \Box(\varphi \alpha)) \rightarrow \Box(\exists \alpha. \varphi \alpha) \text{ in } v]$ 

```


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

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

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

qed

lemma *sign-S5-thm-5*[PLM]:

$[(\Box(\Box(\forall \alpha. \varphi \alpha \equiv \psi \alpha)) \ \& \ (\Box(\forall \alpha. \psi \alpha \equiv \chi \alpha))) \rightarrow (\Box(\forall \alpha. \varphi \alpha \equiv \chi \alpha))] \text{ in } v]$

proof (rule CP)

assume $[\Box(\forall \alpha. \varphi \alpha \equiv \psi \alpha) \ \& \ \Box(\forall \alpha. \psi \alpha \equiv \chi \alpha) \text{ in } v]$

hence $[\Box((\forall \alpha. \varphi \alpha \equiv \psi \alpha) \ \& \ (\forall \alpha. \psi \alpha \equiv \chi \alpha)) \text{ in } v]$

using *KBasic-3*[equiv-rl] **by** *blast*

moreover {

fix v

assume $[(\forall \alpha. \varphi \alpha \equiv \psi \alpha) \ \& \ (\forall \alpha. \psi \alpha \equiv \chi \alpha) \text{ in } v]$

hence $[(\forall \alpha. \varphi \alpha \equiv \chi \alpha) \text{ in } v]$

using *cqt-basic-10*[deduction] **by** *blast*

}

ultimately show $[\Box(\forall \alpha. \varphi \alpha \equiv \chi \alpha) \text{ in } v]$

using *RN-2* **by** *blast*

qed

lemma *id-nec2-1*[PLM]:

$[\Diamond((\alpha::'a::id-eq) = \beta) \equiv (\alpha = \beta) \text{ in } v]$

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*:

$[(\Diamond \varphi) \equiv \psi \text{ in } v] \implies [(\neg \psi) \equiv \Box(\neg \varphi) \text{ in } v]$

proof –

assume $[(\Diamond \varphi) \equiv \psi \text{ in } v]$

moreover have $\bigwedge \varphi \psi. [(\neg \varphi) \equiv \psi \text{ in } v] \implies [(\neg \psi) \equiv \varphi \text{ in } v]$

by *PLM-solver*

ultimately show *?thesis*

unfolding *diamond-def* **by** *blast*

qed

lemma *id-nec2-2*[PLM]:

$[(\alpha::'a::id-eq) \neq \beta] \equiv \Box(\alpha \neq \beta) \text{ in } v]$

using *id-nec2-1*[THEN *id-nec2-2-Aux*] **by** *auto*

lemma *id-nec2-3*[PLM]:

$[(\Diamond((\alpha::'a::id-eq) \neq \beta)) \equiv (\alpha \neq \beta) \text{ in } v]$

using *T* $\Diamond \equiv I$ *id-nec2-2*[equiv-lr]

CP *derived-S5-rules-2-b* **by** *metis*

lemma *exists-desc-box-1*[PLM]:

$[(\exists y. (y^P) = (\iota x. \varphi x)) \rightarrow (\exists y. \Box((y^P) = (\iota x. \varphi x))) \text{ in } v]$

proof (rule CP)

assume $[\exists y. (y^P) = (\iota x. \varphi x) \text{ in } v]$

then obtain y **where** $[(y^P) = (\iota x. \varphi x) \text{ in } v]$

by (rule $\exists E$)

hence $[\Box(y^P = (\iota x. \varphi x)) \text{ in } v]$

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 $[\exists y. \Box((y^P) = (\iota x. \varphi x)) \text{ in } v]$
by (*rule* $\exists I$)
qed

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

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

lemma *en-eq-2*[*PLM*]:
 $[\{x, F\} \equiv \Box \{x, F\} \text{ in } v]$
using *encoding[axiom-instance]* *qml-2[axiom-instance]* **by** (*rule* $\equiv I$)

lemma *en-eq-3*[*PLM*]:
 $[\Diamond \{x, F\} \equiv \{x, F\} \text{ in } v]$
using *encoding[axiom-instance]* *derived-S5-rules-2-b* $\equiv I$ *T \Diamond* **by auto**

lemma *en-eq-4*[*PLM*]:
 $[(\{x, F\} \equiv \{y, G\}) \equiv (\Box \{x, F\} \equiv \Box \{y, G\}) \text{ in } v]$
by (*metis CP en-eq-2* $\equiv I \equiv E(1) \equiv E(2)$)

lemma *en-eq-5*[*PLM*]:
 $[(\Box \{x, F\} \equiv \{y, G\}) \equiv (\Box \{x, F\} \equiv \Box \{y, G\}) \text{ in } v]$
using $\equiv I$ *KBasic-6 encoding[axiom-necessitation, axiom-instance]*
sc-eq-box-box-3[deduction] $\&I$ **by simp**

lemma *en-eq-6*[*PLM*]:
 $[(\{x, F\} \equiv \{y, G\}) \equiv \Box(\{x, F\} \equiv \{y, G\}) \text{ in } v]$
using *en-eq-4 en-eq-5 oth-class-taut-4-a* $\equiv E(6)$ **by meson**

lemma *en-eq-7*[*PLM*]:
 $[(\neg \{x, F\}) \equiv \Box(\neg \{x, F\}) \text{ in } v]$
using *en-eq-3[THEN id-nec2-2-Aux]* **by blast**

lemma *en-eq-8*[*PLM*]:
 $[\Diamond(\neg \{x, F\}) \equiv (\neg \{x, F\}) \text{ in } v]$
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*]:
 $[\Diamond(\neg \{x, F\}) \equiv \Box(\neg \{x, F\}) \text{ in } v]$
using *en-eq-8 en-eq-7* $\equiv E(5)$ **by blast**

lemma *en-eq-10*[*PLM*]:
 $[\mathcal{A}\{x, F\} \equiv \{x, F\} \text{ in } v]$
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  $\varphi$ 
    and IsPropositionalInX  $\psi$ 
    and  $\bigwedge x. [\varphi (x^P) \equiv \psi (x^P) \text{ in } v]$ 
  shows  $[(\lambda y. \varphi (y^P), x^P) \equiv (\lambda y. \psi (y^P), x^P) \text{ in } v]$ 
  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  $\varphi$ 
    and IsPropositionalInXY  $\psi$ 
    and  $\bigwedge x y. [\varphi (x^P) (y^P) \equiv \psi (x^P) (y^P) \text{ in } v]$ 
  shows  $[(\lambda^2 (\lambda x y. \varphi (x^P) (y^P)), x^P, y^P) \equiv (\lambda^2 (\lambda x y. \psi (x^P) (y^P)), x^P, y^P) \text{ in } v]$ 
  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  $\varphi$ 
    and IsPropositionalInXYZ  $\psi$ 
    and  $\bigwedge x y z. [\varphi (x^P) (y^P) (z^P) \equiv \psi (x^P) (y^P) (z^P) \text{ in } v]$ 
  shows  $[(\lambda^3 (\lambda x y z. \varphi (x^P) (y^P) (z^P)), x^P, y^P, z^P) \equiv (\lambda^3 (\lambda x y z. \psi (x^P) (y^P) (z^P)), x^P, y^P, z^P) \text{ in } v]$ 
  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  $\varphi$ 
    and IsPropositionalInX  $\psi$ 
  shows  $[(\Box (\forall x. \varphi (x^P) \equiv \psi (x^P))) \rightarrow (\Box (\forall x. (\lambda y. \varphi (y^P), x^P) \equiv (\lambda y. \psi (y^P), x^P))) \text{ in } v]$ 
  apply (rule qml-1[axiom-instance, deduction])
  apply (rule RN)
  proof (rule CP, rule  $\forall I$ )
  fix  $v x$ 
  assume  $[\forall x. \varphi (x^P) \equiv \psi (x^P) \text{ in } v]$ 
  hence  $\bigwedge x. [\varphi (x^P) \equiv \psi (x^P) \text{ in } v]$ 
  by PLM-solver
  thus  $[(\lambda y. \varphi (y^P), x^P) \equiv (\lambda y. \psi (y^P), x^P) \text{ in } v]$ 
  using assms beta-equiv-eq-1-1 by auto
qed

lemma beta-equiv-eq-2-2[PLM]:
  assumes IsPropositionalInXY  $\varphi$ 
    and IsPropositionalInXY  $\psi$ 
  shows  $[(\Box (\forall x y. \varphi (x^P) (y^P) \equiv \psi (x^P) (y^P))) \rightarrow (\Box (\forall x y. (\lambda^2 (\lambda x y. \varphi (x^P) (y^P)), x^P, y^P) \equiv (\lambda^2 (\lambda x y. \psi (x^P) (y^P)), x^P, y^P))) \text{ in } v]$ 
  apply (rule qml-1[axiom-instance, deduction])

```

apply (*rule RN*)
proof (*rule CP, rule $\forall I$, rule $\forall I$*)
 fix $v\ x\ y$
assume $[\forall x\ y. \varphi(x^P)(y^P) \equiv \psi(x^P)(y^P) \text{ in } v]$
hence $(\bigwedge x\ y. [\varphi(x^P)(y^P) \equiv \psi(x^P)(y^P) \text{ in } v])$
 by (*meson $\forall E$*)
thus $[(\lambda^2(\lambda x\ y. \varphi(x^P)(y^P)), x^P, y^P)]$
 $\equiv (\lambda^2(\lambda x\ y. \psi(x^P)(y^P)), x^P, y^P) \text{ in } v]$
using *assms beta-equiv-eq-1-2* **by** *auto*
qed

lemma *beta-equiv-eq-2-3[PLM]*:
assumes *IsPropositionalInXYZ φ*
and *IsPropositionalInXYZ ψ*
shows $[(\Box(\forall x\ y\ z. \varphi(x^P)(y^P)(z^P) \equiv \psi(x^P)(y^P)(z^P))) \rightarrow$
 $(\Box(\forall x\ y\ z. (\lambda^3(\lambda x\ y\ z. \varphi(x^P)(y^P)(z^P)), x^P, y^P, z^P)]$
 $\equiv (\lambda^3(\lambda x\ y\ z. \psi(x^P)(y^P)(z^P)), x^P, y^P, z^P) \text{ in } v]$
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 $[\forall x\ y\ z. \varphi(x^P)(y^P)(z^P) \equiv \psi(x^P)(y^P)(z^P) \text{ in } v]$
hence $(\bigwedge x\ y\ z. [\varphi(x^P)(y^P)(z^P) \equiv \psi(x^P)(y^P)(z^P) \text{ in } v])$
 by (*meson $\forall E$*)
thus $[(\lambda^3(\lambda x\ y\ z. \varphi(x^P)(y^P)(z^P)), x^P, y^P, z^P)]$
 $\equiv (\lambda^3(\lambda x\ y\ z. \psi(x^P)(y^P)(z^P)), x^P, y^P, z^P) \text{ in } v]$
using *assms beta-equiv-eq-1-3* **by** *auto*
qed

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

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

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

lemma *relations-1[PLM]*:
assumes *IsPropositionalInX φ*
shows $[\exists F. \Box(\forall x. (F, x^P) \equiv \varphi(x^P)) \text{ in } v]$
using *assms* **apply** *cut-tac* **by** *PLM-solver*

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

lemma *relations-3*[PLM]:
assumes *IsPropositionalInXYZ* φ
shows $[\exists F. \Box(\forall x y z. \langle F, x^P, y^P, z^P \rangle \equiv \varphi (x^P) (y^P) (z^P)) \text{ in } v]$
using *assms apply cut-tac by PLM-solver*

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

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

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

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

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

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

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

lemma *thm-relation-negation-2-1*[PLM]:
 $[(\neg \langle F^-, x^P \rangle) \equiv \langle F, x^P \rangle \text{ in } v]$
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]:
 $[(\neg \langle F^-, x^P, y^P \rangle) \equiv \langle F, x^P, y^P \rangle \text{ in } v]$
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]:
 $[(\neg \langle F^-, x^P, y^P, z^P \rangle) \equiv \langle F, x^P, y^P, z^P \rangle \text{ in } v]$
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]:
 $[(p)^- \equiv \neg p \text{ in } v]$
unfolding *propnot-defs*
using *propositions-lemma-2* **by** *simp*

lemma *thm-relation-negation-4*[PLM]:
 $[(\neg((p::o)^-)) \equiv p \text{ in } v]$
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]:
 $[(F::\Pi_1) \neq (F^-) \text{ in } v]$
using *id-eq-prop-prop-2*[deduction]
l-identity[**where** $\varphi=\lambda G . \langle G, x^P \rangle \equiv \langle F^-, x^P \rangle$, *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]:
 $[(F::\Pi_2) \neq (F^-) \text{ in } v]$
using *id-eq-prop-prop-5-a*[deduction]
l-identity[**where** $\varphi=\lambda G . \langle G, x^P, y^P \rangle \equiv \langle F^-, x^P, y^P \rangle$, *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]:
 $[(F::\Pi_3) \neq (F^-) \text{ in } v]$
using *id-eq-prop-prop-5-b*[deduction]
l-identity[**where** $\varphi=\lambda G . \langle G, x^P, y^P, z^P \rangle \equiv \langle F^-, x^P, y^P, z^P \rangle$,
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]:
 $[(p::o) \neq (p^-) \text{ in } v]$
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]:
 $[(p::o)^- = \neg p \text{ in } v]$
unfolding *propnot-defs* **using** *propositions-lemma-1* **by** *simp*

lemma *thm-relation-negation-8*[PLM]:
 $[(p::o) \neq \neg p \text{ in } v]$
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]:
 $[(p::o) = q \rightarrow ((\neg p) = (\neg q)) \text{ in } v]$
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 $[\Box(\forall x. \langle F, x^P \rangle) \vee \Box(\forall x. \neg \langle F, x^P \rangle)]$ *in v*

unfolding *NonContingent-def Necessary-defs Impossible-defs* .

hence $[\Box(\forall x. \neg \langle F^-, x^P \rangle) \vee \Box(\forall x. \neg \langle F, x^P \rangle)]$ *in v*

apply *cut-tac*

apply (*PLM-subst1-method* $\lambda x . \langle F, x^P \rangle \lambda x . \neg \langle F^-, x^P \rangle$)

using *thm-relation-negation-2-1*[*equiv-sym*] **by** *auto*

hence $[\Box(\forall x. \neg \langle F^-, x^P \rangle) \vee \Box(\forall x. \langle F^-, x^P \rangle)]$ *in v*

apply *cut-tac*

apply (*PLM-subst1-goal-method*

$\lambda \varphi . \Box(\forall x. \neg \langle F^-, x^P \rangle) \vee \Box(\forall x. \varphi x) \lambda x . \neg \langle F, x^P \rangle$)

using *thm-relation-negation-1-1*[*equiv-sym*] **by** *auto*

hence $[\Box(\forall x. \langle F^-, x^P \rangle) \vee \Box(\forall x. \neg \langle F^-, x^P \rangle)]$ *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 $[\Box(\forall x. \neg \langle F^-, x^P \rangle) \vee \Box(\forall x. \langle F^-, x^P \rangle)]$ *in v*

unfolding *NonContingent-def Necessary-defs Impossible-defs*

by (*rule oth-class-taut-3-e*[*equiv-lr*])

hence $[\Box(\forall x. \langle F, x^P \rangle) \vee \Box(\forall x. \langle F^-, x^P \rangle)]$ *in v*

apply *cut-tac*

apply (*PLM-subst1-method* $\lambda x . \neg \langle F^-, x^P \rangle \lambda x . \langle F, x^P \rangle$)

using *thm-relation-negation-2-1* **by** *auto*

hence $[\Box(\forall x. \langle F, x^P \rangle) \vee \Box(\forall x. \neg \langle F, x^P \rangle)]$ *in v*

apply *cut-tac*

apply (*PLM-subst1-method* $\lambda x . \langle F^-, x^P \rangle \lambda x . \neg \langle F, x^P \rangle$)

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*]:

$[Contingent F \equiv \Diamond(\exists x . \langle F, x^P \rangle) \ \& \ \Diamond(\exists x . \neg \langle F, x^P \rangle)]$ *in v*

proof (*rule* $\equiv I$; *rule* *CP*)

assume $[Contingent F]$ *in v*

hence $[\neg(\Box(\forall x. \langle F, x^P \rangle) \vee \Box(\forall x. \neg \langle F, x^P \rangle))]$ *in v*

unfolding *Contingent-def Necessary-defs Impossible-defs* .

hence $[(\neg\Box(\forall x. \langle F, x^P \rangle)) \ \& \ (\neg\Box(\forall x. \neg \langle F, x^P \rangle))]$ *in v*

by (*rule oth-class-taut-6-d*[*equiv-lr*])

hence $[(\Diamond\neg(\forall x. \neg \langle F, x^P \rangle)) \ \& \ (\Diamond\neg(\forall x. \langle F, x^P \rangle))]$ *in v*

using *KBasic2-2*[*equiv-lr*] **&I** **&E** **by** *meson*

```

thus [ $(\Diamond(\exists x. \langle F, x^P \rangle)) \ \& \ (\Diamond(\exists x. \neg \langle F, x^P \rangle))$  in  $v$ ]
  unfolding exists-def apply cut-tac
  apply (PLM-subst1-method  $\lambda x. \langle F, x^P \rangle \ \lambda x. \neg \langle F, x^P \rangle$ )
  using oth-class-taut-4-b by auto
next
assume [ $(\Diamond(\exists x. \langle F, x^P \rangle)) \ \& \ (\Diamond(\exists x. \neg \langle F, x^P \rangle))$  in  $v$ ]
hence [ $(\Diamond(\forall x. \neg \langle F, x^P \rangle)) \ \& \ (\Diamond(\forall x. \langle F, x^P \rangle))$  in  $v$ ]
  unfolding exists-def apply cut-tac
  apply (PLM-subst1-goal-method
     $\lambda \varphi. (\Diamond(\neg(\forall x. \neg \langle F, x^P \rangle)) \ \& \ (\Diamond(\neg(\forall x. \varphi x))) \ \lambda x. \neg \langle F, x^P \rangle$ )
  using oth-class-taut-4-b[equiv-sym] by auto
hence [ $(\neg \Box(\forall x. \langle F, x^P \rangle)) \ \& \ (\neg \Box(\forall x. \neg \langle F, x^P \rangle))$  in  $v$ ]
  using KBasic2-2[equiv-rl] &I &E by meson
hence [ $(\neg(\Box(\forall x. \langle F, x^P \rangle) \vee \Box(\forall x. \neg \langle F, x^P \rangle)))$  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]:
  [ $(\Diamond(\exists x. \langle F, x^P \rangle \ \& \ (\Diamond(\neg \langle F, x^P \rangle)))) \equiv \Diamond(\exists x. ((\neg \langle F, x^P \rangle) \ \& \ \Diamond \langle F, x^P \rangle))$  in  $v$ ]

```

```

proof -
  have [ $(\Diamond(\exists x. \langle F, x^P \rangle \ \& \ (\Diamond(\neg \langle F, x^P \rangle))))$  in  $v$ ]
    = [ $(\exists x. \Diamond(\langle F, x^P \rangle \ \& \ \Diamond(\neg \langle F, x^P \rangle)))$  in  $v$ ]
    using BF $\Diamond$ [deduction] CBF $\Diamond$ [deduction] by fast
  also have ... = [ $\exists x. (\Diamond \langle F, x^P \rangle \ \& \ \Diamond(\neg \langle F, x^P \rangle))$  in  $v$ ]
    apply (PLM-subst1-method
       $\lambda x. \Diamond(\langle F, x^P \rangle \ \& \ \Diamond(\neg \langle F, x^P \rangle))$ 
       $\lambda x. \Diamond \langle F, x^P \rangle \ \& \ \Diamond(\neg \langle F, x^P \rangle)$ 
    using S5Basic-12 by auto
  also have ... = [ $\exists x. \Diamond(\neg \langle F, x^P \rangle) \ \& \ \Diamond \langle F, x^P \rangle$  in  $v$ ]
    apply (PLM-subst1-method
       $\lambda x. \Diamond \langle F, x^P \rangle \ \& \ \Diamond(\neg \langle F, x^P \rangle)$ 
       $\lambda x. \Diamond(\neg \langle F, x^P \rangle) \ \& \ \Diamond \langle F, x^P \rangle$ 
    using oth-class-taut-3-b by auto
  also have ... = [ $\exists x. \Diamond((\neg \langle F, x^P \rangle) \ \& \ \Diamond \langle F, x^P \rangle)$  in  $v$ ]
    apply (PLM-subst1-method
       $\lambda x. \Diamond(\neg \langle F, x^P \rangle) \ \& \ \Diamond \langle F, x^P \rangle$ 
       $\lambda x. \Diamond((\neg \langle F, x^P \rangle) \ \& \ \Diamond \langle F, x^P \rangle)$ 
    using S5Basic-12[equiv-sym] by auto
  also have ... = [ $\Diamond(\exists x. ((\neg \langle F, x^P \rangle) \ \& \ \Diamond \langle F, x^P \rangle))$  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]:
  [ $(\Diamond(\exists x. \langle F, x^P \rangle \ \& \ \Diamond(\neg \langle F, x^P \rangle))) \equiv \Diamond(\exists x. \langle F^-, x^P \rangle \ \& \ \Diamond(\neg \langle F^-, x^P \rangle))$ ]

```

```

in v]
  apply (PLM-subst1-method  $\lambda x . (\llbracket F, x^P \rrbracket \lambda x . \neg(\llbracket F^-, x^P \rrbracket))$ )
  using thm-relation-negation-2-1[equiv-sym] apply simp
  apply (PLM-subst1-method  $\lambda x . \neg(\llbracket F, x^P \rrbracket \lambda x . (\llbracket F^-, x^P \rrbracket))$ )
  using thm-relation-negation-1-1[equiv-sym] apply simp
  using lem-cont-e by simp

lemma thm-cont-e-1[PLM]:
  [ $\Diamond(\exists x . ((\neg(\llbracket E!, x^P \rrbracket)) \ \&\ (\Diamond(\llbracket E!, x^P \rrbracket))))$ ] 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]:
  [ $\exists (F::\Pi_1) G . (F \neq G \ \&\ \text{Contingent } F \ \&\ \text{Contingent } G)$ ] 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  $L \equiv (\lambda x . (\llbracket E!, x^P \rrbracket \rightarrow \llbracket E!, x^P \rrbracket))$ 

  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  $\forall I(1)$ )

  lemma thm-noncont-e-e-4[PLM]:

```

```

[NonContingent ( $L^-$ ) in  $v$ ]
unfolding NonContingent-def using thm-noncont-e-e-2
by (rule  $\forall I(2)$ )

lemma thm-noncont-e-e-5[PLM]:
   $v]$   $[\exists (F::\Pi_1) G . F \neq G \ \& \ NonContingent F \ \& \ NonContingent G \text{ in } v]$ 
    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]:
   $[NonContingent (F::\Pi_1) \rightarrow \neg(\exists G . (Contingent G \ \& \ G = F)) \text{ in } v]$ 
  proof (rule CP)
    assume  $[NonContingent F \text{ in } v]$ 
    hence  $[\neg(Contingent F) \text{ in } v]$ 
    unfolding NonContingent-def Contingent-def
    apply cut-tac by PLM-solver
    moreover {
      assume  $[\exists G . Contingent G \ \& \ G = F \text{ in } v]$ 
      then obtain  $P$  where  $[Contingent P \ \& \ P = F \text{ in } v]$ 
      by (rule  $\exists E$ )
      hence  $[Contingent F \text{ in } v]$ 
      using  $\&E$  l-identity[axiom-instance, deduction, deduction]
      by blast
    }
    ultimately show  $[\neg(\exists G . Contingent G \ \& \ G = F) \text{ in } v]$ 
    using modus-tollens-1 CP by blast
  qed

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

lemma four-distinct-3[PLM]:
   $[L \neq (L^-) \ \& \ L \neq E! \ \& \ L \neq (E!^-) \ \& \ (L^-) \neq E!$ 
     $\ \& \ (L^-) \neq (E!^-) \ \& \ E! \neq (E!^-) \text{ in } v]$ 

```

```

proof (rule &I)+
  show  $[L \neq (L^-) \text{ in } v]$ 
  by (rule thm-relation-negation-5-1)
next
{
  assume  $[L = E! \text{ in } v]$ 
  hence  $[NonContingent\ L \ \& \ L = E! \text{ in } v]$ 
    using thm-noncont-e-e-3 &I by auto
  hence  $[\exists\ G . NonContingent\ G \ \& \ G = E! \text{ in } v]$ 
    using thm-noncont-e-e-3 &I  $\exists I$  by fast
}
thus  $[L \neq E! \text{ in } v]$ 
  using four-distinct-2[deduction, OF thm-cont-e-2]
    modus-tollens-1 CP
  by blast
next
{
  assume  $[L = (E!^-) \text{ in } v]$ 
  hence  $[NonContingent\ L \ \& \ L = (E!^-) \text{ in } v]$ 
    using thm-noncont-e-e-3 &I by auto
  hence  $[\exists\ G . NonContingent\ G \ \& \ G = (E!^-) \text{ in } v]$ 
    using thm-noncont-e-e-3 &I  $\exists I$  by fast
}
thus  $[L \neq (E!^-) \text{ in } v]$ 
  using four-distinct-2[deduction, OF thm-cont-e-3]
    modus-tollens-1 CP
  by blast
next
{
  assume  $[(L^-) = E! \text{ in } v]$ 
  hence  $[NonContingent\ (L^-) \ \& \ (L^-) = E! \text{ in } v]$ 
    using thm-noncont-e-e-4 &I by auto
  hence  $[\exists\ G . NonContingent\ G \ \& \ G = E! \text{ in } v]$ 
    using thm-noncont-e-e-3 &I  $\exists I$  by fast
}
thus  $[(L^-) \neq E! \text{ in } v]$ 
  using four-distinct-2[deduction, OF thm-cont-e-2]
    modus-tollens-1 CP
  by blast
next
{
  assume  $[(L^-) = (E!^-) \text{ in } v]$ 
  hence  $[NonContingent\ (L^-) \ \& \ (L^-) = (E!^-) \text{ in } v]$ 
    using thm-noncont-e-e-4 &I by auto
  hence  $[\exists\ G . NonContingent\ G \ \& \ G = (E!^-) \text{ in } v]$ 
    using thm-noncont-e-e-3 &I  $\exists I$  by fast
}
thus  $[(L^-) \neq (E!^-) \text{ in } v]$ 
  using four-distinct-2[deduction, OF thm-cont-e-3]
    modus-tollens-1 CP
  by blast
next
show  $[E! \neq (E!^-) \text{ in } v]$ 
  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 $[\Box p \vee \Box \neg p\ in\ v]$
 unfolding *NonContingent-def Necessary-defs Impossible-defs* .
 hence $[\Box(\neg(p^-)) \vee \Box(\neg p)\ in\ v]$
 apply *cut-tac*
 apply (*PLM-subst-method* $p\ \neg(p^-)$)
 using *thm-relation-negation-4*[*equiv-sym*] **by** *auto*
 hence $[\Box(\neg(p^-)) \vee \Box(p^-)\ in\ v]$
 apply *cut-tac*
 apply (*PLM-subst-goal-method* $\lambda\varphi . \Box(\neg(p^-)) \vee \Box(\varphi)\ \neg p$)
 using *thm-relation-negation-3*[*equiv-sym*] **by** *auto*
 hence $[\Box(p^-) \vee \Box(\neg(p^-))\ in\ v]$
 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 $[\Box(\neg(p^-)) \vee \Box(p^-)\ in\ v]$
 unfolding *NonContingent-def Necessary-defs Impossible-defs*
 by (*rule* *oth-class-taut-3-e*[*equiv-lr*])
 hence $[\Box(p) \vee \Box(p^-)\ in\ v]$
 apply *cut-tac*
 apply (*PLM-subst-goal-method* $\lambda\varphi . \Box\varphi \vee \Box(p^-)\ \neg(p^-)$)
 using *thm-relation-negation-4* **by** *auto*
 hence $[\Box(p) \vee \Box(\neg p)\ in\ v]$
 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 $[\neg(\Box p \vee \Box(\neg p))\ in\ v]$
 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 $[(\Diamond \neg(\neg p)) \ \& \ (\Diamond \neg(p)) \text{ in } v]$
 apply *cut-tac* apply *PLM-solver*
 apply $(PLM\text{-subst-method } p \ \neg \neg p)$
 using *oth-class-taut-4-b* by *auto*
 hence $[(\neg \Box p) \ \& \ (\neg \Box(\neg p)) \text{ in } v]$
 using *KBasic2-2[equiv-rl]* & *I* & *E* by *meson*
 hence $[\neg(\Box(p) \vee \Box(\neg p)) \text{ in } v]$
 by $(rule \text{ oth-class-taut-6-d[equiv-rl]})$
 thus $[Contingent \ p \text{ in } v]$
 unfolding *Contingent-def Necessary-defs Impossible-defs* .
 qed

lemma *thm-cont-propos-3[PLM]*:
 $[Contingent \ (p::o) \equiv Contingent \ (p^-) \text{ in } v]$
 using *thm-cont-propos-1*
 unfolding *NonContingent-def Contingent-def*
 by $(rule \text{ oth-class-taut-5-d[equiv-lr]})$

context
begin
private definition p_0 **where**
 $p_0 \equiv \forall x. (\Box!, x^P) \rightarrow (\Box!, x^P)$

lemma *thm-noncont-propos-1[PLM]*:
 $[Necessary \ p_0 \text{ in } v]$
 unfolding *Necessary-defs p₀-def*
 apply $(rule \text{ RN, rule } \forall I)$
 using *if-p-then-p* .

lemma *thm-noncont-propos-2[PLM]*:
 $[Impossible \ (p_0^-) \text{ in } v]$
 unfolding *Impossible-defs*
 apply $(PLM\text{-subst-method } \neg p_0 \ p_0^-)$
 using *thm-relation-negation-3[equiv-sym]* apply *simp*
 apply $(PLM\text{-subst-method } p_0 \ \neg \neg p_0)$
 using *oth-class-taut-4-b* apply *simp*
 using *thm-noncont-propos-1* unfolding *Necessary-defs*
 by *simp*

lemma *thm-noncont-propos-3[PLM]*:
 $[NonContingent \ (p_0) \text{ in } v]$
 unfolding *NonContingent-def* using *thm-noncont-propos-1*
 by $(rule \ \forall I(1))$

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

lemma *thm-noncont-propos-5[PLM]*:
 $[\exists \ (p::o) \ q. p \neq q \ \& \ NonContingent \ p \ \& \ NonContingent \ q \text{ in } v]$
 apply $(rule\text{-tac } \alpha = p_0 \text{ in } \exists I, rule\text{-tac } \alpha = p_0^- \text{ in } \exists I)$
 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]:

$[\exists p . \Diamond p \ \& \ \Diamond(\neg p) \text{ in } v]$

apply (*rule-tac* $\alpha=q_0$ **in** $\exists I$) **unfolding** $q_0\text{-def}$

using *qml-4*[*axiom-instance*] **by** *simp*

lemma *basic-prop-2*[PLM]:

$[Contingent\ q_0 \text{ in } v]$

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 $q_0\text{-def diamond-def}$ **by** *simp*

lemma *basic-prop-3*[PLM]:

$[Contingent\ (q_0^-) \text{ in } v]$

apply (*rule thm-cont-propos-3*[*equiv-lr*])

using *basic-prop-2* .

lemma *basic-prop-4*[PLM]:

$[\exists (p::o) \ q . p \neq q \ \& \ Contingent\ p \ \& \ Contingent\ q \text{ in } v]$

apply (*rule-tac* $\alpha=q_0$ **in** $\exists I$, *rule-tac* $\alpha=q_0^-$ **in** $\exists I$)

using *thm-relation-negation-6 basic-prop-2 basic-prop-3* **&I** **by** *simp*

lemma *four-distinct-props-1*[PLM]:

$[NonContingent\ (p::\Pi_0) \rightarrow (\neg(\exists \ q . Contingent\ q \ \& \ q = p)) \text{ in } v]$

proof (*rule CP*)

assume $[NonContingent\ p \text{ in } v]$

hence $[\neg(Contingent\ p) \text{ in } v]$

unfolding *NonContingent-def Contingent-def*

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

moreover {

assume $[\exists \ q . Contingent\ q \ \& \ q = p \text{ in } v]$

then obtain r **where** $[Contingent\ r \ \& \ r = p \text{ in } v]$

by (*rule* $\exists E$)

hence $[Contingent\ p \text{ in } v]$

using *&E l-identity*[*axiom-instance, deduction, deduction*]

by *blast*

}

ultimately show $[\neg(\exists \ q . Contingent\ q \ \& \ q = p) \text{ in } v]$

using *modus-tollens-1 CP* **by** *blast*

qed

lemma *four-distinct-props-2*[PLM]:

$[Contingent\ (p::o) \rightarrow \neg(\exists \ q . (NonContingent\ q \ \& \ q = p)) \text{ in } v]$

proof (*rule CP*)

assume $[Contingent\ p \text{ in } v]$

hence $[\neg(NonContingent\ p) \text{ in } v]$

unfolding *NonContingent-def Contingent-def*

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


```

moreover {
  assume  $[\exists q . \text{NonContingent } q \ \& \ q = p \text{ in } v]$ 
  then obtain  $r$  where  $[\text{NonContingent } r \ \& \ r = p \text{ in } v]$ 
  by (rule  $\exists E$ )
  hence  $[\text{NonContingent } p \text{ in } v]$ 
  using  $\&E$  l-identity [axiom-instance, deduction, deduction]
  by blast
}
ultimately show  $[\neg(\exists q . \text{NonContingent } q \ \& \ q = p) \text{ in } v]$ 
using modus-tollens-1 CP by blast
qed

lemma four-distinct-props-4 [PLM]:
 $[p_0 \neq (p_0^-) \ \& \ p_0 \neq q_0 \ \& \ p_0 \neq (q_0^-) \ \& \ (p_0^-) \neq q_0$ 
 $\ \& \ (p_0^-) \neq (q_0^-) \ \& \ q_0 \neq (q_0^-) \text{ in } v]$ 
proof (rule  $\&I$ ) +
  show  $[p_0 \neq (p_0^-) \text{ in } v]$ 
  by (rule thm-relation-negation-6)
next
  {
    assume  $[p_0 = q_0 \text{ in } v]$ 
    hence  $[\exists q . \text{NonContingent } q \ \& \ q = q_0 \text{ in } v]$ 
    using  $\&I$  thm-noncont-propos-3  $\exists I$  [where  $\alpha=p_0$ ]
    by simp
  }
  thus  $[p_0 \neq q_0 \text{ in } v]$ 
  using four-distinct-props-2 [deduction, OF basic-prop-2]
  modus-tollens-1 CP
  by blast
next
  {
    assume  $[p_0 = (q_0^-) \text{ in } v]$ 
    hence  $[\exists q . \text{NonContingent } q \ \& \ q = (q_0^-) \text{ in } v]$ 
    using thm-noncont-propos-3  $\&I \exists I$  [where  $\alpha=p_0$ ] by simp
  }
  thus  $[p_0 \neq (q_0^-) \text{ in } v]$ 
  using four-distinct-props-2 [deduction, OF basic-prop-3]
  modus-tollens-1 CP
  by blast
next
  {
    assume  $[(p_0^-) = q_0 \text{ in } v]$ 
    hence  $[\exists q . \text{NonContingent } q \ \& \ q = q_0 \text{ in } v]$ 
    using thm-noncont-propos-4  $\&I \exists I$  [where  $\alpha=p_0^-$ ] by auto
  }
  thus  $[(p_0^-) \neq q_0 \text{ in } v]$ 
  using four-distinct-props-2 [deduction, OF basic-prop-2]
  modus-tollens-1 CP
  by blast
next
  {
    assume  $[(p_0^-) = (q_0^-) \text{ in } v]$ 
    hence  $[\exists q . \text{NonContingent } q \ \& \ q = (q_0^-) \text{ in } v]$ 
    using thm-noncont-propos-4  $\&I \exists I$  [where  $\alpha=p_0^-$ ] by auto
  }

```

```

}
thus [( $p_0^-$ )  $\neq$  ( $q_0^-$ ) in  $v$ ]
  using four-distinct-props-2[deduction, OF basic-prop-3]
      modus-tollens-1 CP
  by blast
next
  show [ $q_0 \neq (q_0^-)$  in  $v$ ]
    by (rule thm-relation-negation-6)
qed

lemma cont-true-cont-1[PLM]:
  [ContingentlyTrue  $p \rightarrow$  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◇[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  $[\neg q_0 \text{ in } v]$ 
  hence  $[(\neg q_0) \ \&\ \Diamond q_0 \text{ in } v]$ 
    unfolding  $q_0\text{-def}$ 
    using  $qml\text{-}4[axiom\text{-}instance, conj1] \ \&I$ 
    by auto
}
ultimately show ?thesis
  unfolding  $ContingentlyTrue\text{-}def \ ContingentlyFalse\text{-}def$ 
  using  $\vee E(4) \ CP$  by auto
qed

```

```

lemma  $cont\text{-}tf\text{-}thm\text{-}2[PLM]:$ 
 $[ContingentlyFalse \ q_0 \ \vee \ ContingentlyFalse \ (q_0^-) \text{ in } v]$ 
using  $cont\text{-}tf\text{-}thm\text{-}1 \ cont\text{-}true\text{-}cont\text{-}3[\textbf{where } p=q_0]$ 
 $cont\text{-}true\text{-}cont\text{-}4[\textbf{where } p=q_0]$ 
apply cut-tac by PLM-solver

```

```

lemma  $cont\text{-}tf\text{-}thm\text{-}3[PLM]:$ 
 $[\exists \ p . \ ContingentlyTrue \ p \text{ in } v]$ 
proof  $(rule \ \vee E(1); (rule \ CP)?)$ 
  show  $[ContingentlyTrue \ q_0 \ \vee \ ContingentlyFalse \ q_0 \text{ in } v]$ 
    using  $cont\text{-}tf\text{-}thm\text{-}1$  .
next
  assume  $[ContingentlyTrue \ q_0 \text{ in } v]$ 
  thus ?thesis
    using  $\exists I$  by metis
next
  assume  $[ContingentlyFalse \ q_0 \text{ in } v]$ 
  hence  $[ContingentlyTrue \ (q_0^-) \text{ in } v]$ 
    using  $cont\text{-}true\text{-}cont\text{-}4[equiv\text{-}lr]$  by simp
  thus ?thesis
    using  $\exists I$  by metis
qed

```

```

lemma  $cont\text{-}tf\text{-}thm\text{-}4[PLM]:$ 
 $[\exists \ p . \ ContingentlyFalse \ p \text{ in } v]$ 
proof  $(rule \ \vee E(1); (rule \ CP)?)$ 
  show  $[ContingentlyTrue \ q_0 \ \vee \ ContingentlyFalse \ q_0 \text{ in } v]$ 
    using  $cont\text{-}tf\text{-}thm\text{-}1$  .
next
  assume  $[ContingentlyTrue \ q_0 \text{ in } v]$ 
  hence  $[ContingentlyFalse \ (q_0^-) \text{ in } v]$ 
    using  $cont\text{-}true\text{-}cont\text{-}3[equiv\text{-}lr]$  by simp
  thus ?thesis
    using  $\exists I$  by metis
next
  assume  $[ContingentlyFalse \ q_0 \text{ in } v]$ 
  thus ?thesis
    using  $\exists I$  by metis
qed

```

```

lemma  $cont\text{-}tf\text{-}thm\text{-}5[PLM]:$ 

```

```

[ContingentlyTrue p & Necessary q  $\rightarrow$  p  $\neq$  q in v]
proof (rule CP)
  assume [ContingentlyTrue p & Necessary q in v]
  hence 1: [ $\Diamond(\neg p)$  &  $\Box q$  in v]
    unfolding ContingentlyTrue-def Necessary-defs
    using &E &I by blast
  hence [ $\neg\Box p$  in v]
    apply cut-tac apply (drule &E(1))
    unfolding diamond-def
    apply (PLM-subst-method  $\neg\neg p$ )
    using oth-class-taut-4-b[equiv-sym] by auto
  moreover {
    assume [p = q in v]
    hence [ $\Box p$  in v]
      using l-identity[where  $\alpha=q$  and  $\beta=p$  and  $\varphi=\lambda x . \Box x$ ,
        axiom-instance, deduction, deduction]
        1[conj2] id-eq-prop-prop-8-b[deduction]
      by blast
    }
  ultimately show [p  $\neq$  q in v]
    using modus-tollens-1 CP by blast
qed

```

```

lemma cont-tf-thm-6[PLM]:
  [(ContingentlyFalse p & Impossible q)  $\rightarrow$  p  $\neq$  q in v]
  proof (rule CP)
    assume [ContingentlyFalse p & Impossible q in v]
    hence 1: [ $\Diamond p$  &  $\Box(\neg q)$  in v]
      unfolding ContingentlyFalse-def Impossible-defs
      using &E &I by blast
    hence [ $\neg\Diamond q$  in v]
      unfolding diamond-def apply cut-tac by PLM-solver
    moreover {
      assume [p = q in v]
      hence [ $\Diamond q$  in v]
        using l-identity[axiom-instance, deduction, deduction] 1[conj1]
          id-eq-prop-prop-8-b[deduction]
        by blast
      }
    ultimately show [p  $\neq$  q in v]
      using modus-tollens-1 CP by blast
  qed
end

```

```

lemma oa-contingent-1[PLM]:
  [O!  $\neq$  A! in v]
  proof -
    {
      assume [O! = A! in v]
      hence [( $\lambda x . \Diamond(E!, x^P)$ ) = ( $\lambda x . \neg\Diamond(E!, x^P)$ ) in v]
        unfolding Ordinary-def Abstract-def .
      moreover have [( $\lambda x . \Diamond(E!, x^P)$ ),  $x^P$ ]  $\equiv$   $\Diamond(E!, x^P)$  in v]
        apply (rule beta-C-meta-1) by (rule IsPropositional-intros)+
      ultimately have [( $\lambda x . \neg\Diamond(E!, x^P)$ ),  $x^P$ ]  $\equiv$   $\Diamond(E!, x^P)$  in v]

```

```

    using l-identity[axiom-instance, deduction, deduction] by fast
    moreover have  $[(\lambda x. \neg \Diamond(E!, x^P)), x^P] \equiv \neg \Diamond(E!, x^P)$  in v]
    apply (rule beta-C-meta-1) by (rule IsPropositional-intros)+
    ultimately have  $[\Diamond(E!, x^P) \equiv \neg \Diamond(E!, x^P)]$  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]:
 $[(\Diamond O!, x^P) \equiv \neg \Diamond A!, x^P]$ in v]
proof –
 have $[(\lambda x. \neg \Diamond(E!, x^P)), x^P] \equiv \neg \Diamond(E!, x^P)$ in v]
 apply (rule beta-C-meta-1)
 by (rule IsPropositional-intros)+
 hence $[(\neg (\lambda x. \neg \Diamond(E!, x^P)), x^P) \equiv \Diamond(E!, x^P)]$ in v]
 using oth-class-taut-5-d[equiv-lr] oth-class-taut-4-b[equiv-sym]
 $\equiv E(5)$ by blast
 moreover have $[(\lambda x. \Diamond(E!, x^P)), x^P] \equiv \Diamond(E!, x^P)$ 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]:
 $[(\Diamond A!, x^P) \equiv \neg \Diamond O!, x^P]$ 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 $\lambda x. \Diamond(E!, x^P) \lambda x. (\lambda x. \Diamond(E!, x^P), x^P)$)
 apply (rule beta-C-meta-1[equiv-sym]; (rule IsPropositional-intros)+)
 using BF \Diamond [deduction, OF thm-cont-prop-2[equiv-lr, OF thm-cont-e-2,
 conj1]]
 by (rule T \Diamond [deduction])
 subgoal
 apply (PLM-subst1-method $\lambda x. (\Diamond A!, x^P) \lambda x. \neg \Diamond O!, x^P)$)
 using oa-contingent-3 apply simp
 using cqt-further-5[deduction, conj1, OF A-objects[axiom-instance]]
 by (rule T \Diamond [deduction])
 done

lemma *oa-contingent-5*[PLM]:
 [Contingent A! in v]
 apply (rule thm-cont-prop-2[equiv-rl], rule &I)
 subgoal

```

    using cqt-further-5[deduction,conj1, OF A-objects[axiom-instance]]
    by (rule T◇[deduction])
  subgoal
    unfolding Abstract-def
    apply (PLM-subst1-method  $\lambda x. \neg \Diamond \langle E!, x^P \rangle \lambda x. \langle \lambda x. \neg \Diamond \langle E!, x^P \rangle, x^P \rangle$ )
    apply (rule beta-C-meta-1[equiv-sym]; (rule IsPropositional-intros)+)
    apply (PLM-subst1-method  $\lambda x. \Diamond \langle E!, x^P \rangle \lambda x. \neg \neg \Diamond \langle E!, x^P \rangle$ )
    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]:
   $[(O!^-) \neq (A!^-) \text{ in } v]$ 
proof -
  {
    assume  $[(O!^-) = (A!^-) \text{ in } v]$ 
    hence  $[(\lambda x. \neg \langle O!, x^P \rangle) = (\lambda x. \neg \langle A!, x^P \rangle) \text{ in } v]$ 
      unfolding propnot-defs .
    moreover have  $[(\langle \lambda x. \neg \langle O!, x^P \rangle, x^P \rangle \equiv \neg \langle O!, x^P \rangle) \text{ in } v]$ 
      apply (rule beta-C-meta-1)
      by (rule IsPropositional-intros)+
    ultimately have  $[(\langle \lambda x. \neg \langle A!, x^P \rangle, x^P \rangle \equiv \neg \langle O!, x^P \rangle) \text{ in } v]$ 
      using l-identity[axiom-instance, deduction, deduction]
      by fast
    hence  $[(\neg \langle A!, x^P \rangle) \equiv \neg \langle O!, x^P \rangle \text{ in } v]$ 
      apply cut-tac
      apply (PLM-subst-method  $\langle \lambda x. \neg \langle A!, x^P \rangle, x^P \rangle (\neg \langle A!, x^P \rangle)$ )
      apply (rule beta-C-meta-1; (rule IsPropositional-intros)+)
      by assumption
    hence  $[(\langle O!, x^P \rangle \equiv \neg \langle O!, x^P \rangle) \text{ in } v]$ 
      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]:
   $[(\langle O!^-, x^P \rangle \equiv \neg \langle A!^-, x^P \rangle) \text{ in } v]$ 
proof -
  have  $[(\neg \langle \lambda x. \neg \langle A!, x^P \rangle, x^P \rangle) \equiv \langle A!, x^P \rangle \text{ in } v]$ 
    apply (PLM-subst-method  $(\neg \langle A!, x^P \rangle) \langle \lambda x. \neg \langle A!, x^P \rangle, x^P \rangle$ )
    apply (rule beta-C-meta-1[equiv-sym];
      (rule IsPropositional-intros)+)
    using oth-class-taut-4-b[equiv-sym] by auto
  moreover have  $[(\langle \lambda x. \neg \langle O!, x^P \rangle, x^P \rangle \equiv \neg \langle O!, x^P \rangle) \text{ in } v]$ 
    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]:
 $[Contingent (O!)^- \text{ in } v]$
using *oa-contingent-4 thm-cont-prop-3*[equiv-lr] **by** *auto*

lemma *oa-contingent-9*[PLM]:
 $[Contingent (A!)^- \text{ in } v]$
using *oa-contingent-5 thm-cont-prop-3*[equiv-lr] **by** *auto*

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

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

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

```

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

lemma oa-facts-5[PLM]:
  [ $\Diamond(O!, x^P) \equiv \Box(O!, x^P)$  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]:
  [ $\Diamond(A!, x^P) \equiv \Box(A!, x^P)$  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  $\equiv I$ ; 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 ( $\lambda x. \Diamond(E!, x^P), x^P$ )  $\Diamond(E!, x^P)$ )
    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  $\Diamond(E!, x^P)$  ( $\lambda x. \Diamond(E!, x^P), x^P$ ))
    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  $\equiv I$ ; 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 ( $\lambda x. \neg\Diamond(E!, x^P), x^P$ )  $\neg\Diamond(E!, x^P)$ )
    by (rule beta-C-meta-1, (rule IsPropositional-intros | assumption))+
    hence [ $\mathcal{A}(\Box\neg\Diamond(E!, x^P))$  in v]
    apply cut-tac
    apply (PLM-subst-method ( $\neg\Diamond(E!, x^P)$ ) ( $\Box\neg\Diamond(E!, x^P)$ ))
    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 [ $\langle A!, x^P \rangle$  in  $v$ ]
  unfolding Abstract-def apply cut-tac
  apply (PLM-subst-method  $\neg \langle E!, x^P \rangle$  ( $\langle \lambda x. \neg \langle E!, x^P \rangle, x^P \rangle$ )
    by (rule beta-C-meta-1[equiv-sym], (rule IsPropositional-intros |
assumption)+)
  qed

lemma cont-nec-fact1-1[PLM]:
  [WeaklyContingent  $F \equiv \text{WeaklyContingent } (F^-)$  in  $v$ ]
  proof (rule  $\equiv I$ ; rule CP)
    assume [WeaklyContingent  $F$  in  $v$ ]
    hence wc-def: [Contingent  $F \ \& \ (\forall x. (\langle F, x^P \rangle \rightarrow \Box \langle F, x^P \rangle))$  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 [ $\langle F^-, x^P \rangle$  in  $v$ ]
          hence [ $\neg \Box \langle F, x^P \rangle$  in  $v$ ]
            unfolding diamond-def apply cut-tac
            apply (PLM-subst-method  $\neg \langle F^-, x^P \rangle$  ( $\langle F, x^P \rangle$ )
              using thm-relation-negation-2-1 by auto
            moreover {
              assume [ $\neg \Box \langle F^-, x^P \rangle$  in  $v$ ]
              hence [ $\neg \Box \langle \lambda x. \neg \langle F, x^P \rangle, x^P \rangle$  in  $v$ ]
                unfolding propnot-defs .
                hence [ $\langle F, x^P \rangle$  in  $v$ ]
                unfolding diamond-def
                apply cut-tac apply (PLM-subst-method ( $\langle \lambda x. \neg \langle F, x^P \rangle, x^P \rangle$ )
   $\neg \langle F, x^P \rangle$ )
                  apply (rule beta-C-meta-1; rule IsPropositional-intros)
                  by simp
                hence [ $\Box \langle F, x^P \rangle$  in  $v$ ]
                using wc-def[conj2] cqt-1[axiom-instance, deduction]
                modus-ponens by fast
              }
            ultimately have [ $\Box \langle F^-, x^P \rangle$  in  $v$ ]
              using  $\neg \neg E$  modus-tollens-1 CP by blast
            }
          hence [ $\forall x. \langle F^-, x^P \rangle \rightarrow \Box \langle F^-, x^P \rangle$  in  $v$ ]
            using  $\forall I$  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  $(F^-) \ \& \ (\forall x. (\langle F^-, x^P \rangle \rightarrow \Box \langle F^-, x^P \rangle))$ 
  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 [ $\Diamond(F, x^P)$ ] in v
  hence [ $\neg \Box(F^-, x^P)$ ] in v
    unfolding diamond-def apply cut-tac
    apply (PLM-subst-method  $\neg(F, x^P) \Diamond(F^-, x^P)$ )
    using thm-relation-negation-1-1[equiv-sym] by auto
  moreover {
    assume [ $\neg \Box(F, x^P)$ ] in v
    hence [ $\Diamond(F^-, x^P)$ ] in v
      unfolding diamond-def
      apply cut-tac apply (PLM-subst-method  $\Diamond(F, x^P) \neg(F^-, x^P)$ )
      using thm-relation-negation-2-1[equiv-sym] by auto
    hence [ $\Box(F^-, x^P)$ ] in v
      using wc-def[conj2] cqt-1[axiom-instance, deduction]
      modus-ponens by fast
  }
  ultimately have [ $\Box(F, x^P)$ ] in v
    using  $\neg\neg E$  modus-tollens-1 CP by blast
}
hence [ $\forall x. \Diamond(F, x^P) \rightarrow \Box(F, x^P)$ ] in v
  using  $\forall I$  CP by fast
}
ultimately show [WeaklyContingent (F) in v]
  unfolding WeaklyContingent-def by (rule &I)
qed

```

lemma *cont-nec-fact1-2[PLM]*:
 $[(\text{WeaklyContingent } F \ \& \ \neg(\text{WeaklyContingent } G)) \rightarrow (F \neq G)]$ in v
 using l-identity[axiom-instance, deduction, deduction] &E &I
 modus-tollens-1 CP by metis

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

lemma *cont-nec-fact2-2[PLM]*:
 $[\text{WeaklyContingent } (A!)]$ in v
 unfolding WeaklyContingent-def
 apply (rule &I)
 using oa-contingent-5 apply simp
 using oa-facts-6 unfolding equiv-def
 using &E(1) $\forall I$ by fast

lemma *cont-nec-fact2-3[PLM]*:
 $[\neg(\text{WeaklyContingent } (E!))]$ in v
proof (rule modus-tollens-1, rule CP)
 assume [$\text{WeaklyContingent } E!$] in v
 thus [$\forall x. \Diamond(E!, x^P) \rightarrow \Box(E!, x^P)$] in v
 unfolding WeaklyContingent-def using &E(2) by fast

```

next
{
  assume 1:  $[\forall x . \Diamond(\Box(E!, x^P)) \rightarrow \Box(\Box(E!, x^P)) \text{ in } v]$ 
  have  $[\exists x . \Diamond(\Box(E!, x^P) \ \&\ \Diamond(\neg(\Box(E!, x^P)))) \text{ in } v]$ 
    using qml-4[axiom-instance, conj1, THEN BFs-3[deduction]] .
  then obtain  $x$  where  $[\Diamond(\Box(E!, x^P) \ \&\ \Diamond(\neg(\Box(E!, x^P)))) \text{ in } v]$ 
    by (rule  $\exists E$ )
  hence  $[\Diamond(\Box(E!, x^P) \ \&\ \Diamond(\neg(\Box(E!, x^P)))) \text{ in } v]$ 
    using KBasic2-8[deduction] S5Basic-8[deduction]
    &I &E by blast
  hence  $[\Box(\Box(E!, x^P) \ \&\ \neg\Box(E!, x^P)) \text{ in } v]$ 
    using 1[THEN  $\forall E$ , deduction] &E &I
    KBasic2-2[equiv-rl] by blast
  hence  $[\neg(\forall x . \Diamond(\Box(E!, x^P)) \rightarrow \Box(\Box(E!, x^P))) \text{ in } v]$ 
    using oth-class-taut-1-a modus-tollens-1 CP by blast
}
thus  $[\neg(\forall x . \Diamond(\Box(E!, x^P)) \rightarrow \Box(\Box(E!, x^P))) \text{ in } v]$ 
  using reductio-aa-2 if-p-then-p CP by meson
qed

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

```

```

      thus  $[O! \neq (PLM.L^-) \text{ in } v]$ 
      using cont-nec-fact2-1 cont-nec-fact1-2[deduction] &I by simp
qed

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

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

lemma id-nec3-2[PLM]:
 $[\Diamond((x^P) =_E (y^P)) \equiv ((x^P) =_E (y^P)) \text{ in } v]$ 

```

```

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

```

```

lemma thm-neg-eqE[PLM]:
 $[\neg((x^P) \neq_E (y^P)) \equiv \neg((x^P) =_E (y^P))] \text{ in } v]$ 
proof –
  have  $[(x^P) \neq_E (y^P) \text{ in } v] = [\neg(\lambda^2 (\lambda x y . (x^P) =_E (y^P)))^-, x^P,$ 
 $y^P] \text{ in } v]$ 
    unfolding not-identicalE-def by simp
  also have  $... = [\neg(\lambda^2 (\lambda x y . (x^P) =_E (y^P))), x^P, y^P] \text{ in } v]$ 
    unfolding propnot-defs using beta-C-meta-2[equiv-lr]
      beta-C-meta-2[equiv-rl] IsPropositional-intros by fast
  also have  $... = [\neg((x^P) =_E (y^P)) \text{ in } v]$ 
    apply (PLM-subst-method
       $(\lambda^2 (\lambda x y . (x^P) =_E (y^P))), x^P, y^P]$ 
       $(x^P) =_E (y^P))$ 
    apply (rule beta-C-meta-2) unfolding identity-defs
    apply (rule IsPropositional-intros)
    by auto
  finally show ?thesis
    using  $\equiv I$  CP by presburger
qed

```

```

lemma id-nec4-1[PLM]:
 $[\neg((x^P) \neq_E (y^P)) \equiv \Box((x^P) \neq_E (y^P))] \text{ in } v]$ 
proof –
  have  $[\neg((x^P) =_E (y^P))] \equiv \Box(\neg((x^P) =_E (y^P))) \text{ in } v]$ 
    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  $(\neg((x^P) =_E (y^P))) (x^P) \neq_E (y^P))$ 
      using thm-neg-eqE[equiv-sym] by auto
qed

```

```

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

```

```

lemma id-act-1[PLM]:
 $[(x^P) =_E (y^P)] \equiv (\mathcal{A}((x^P) =_E (y^P))) \text{ in } v]$ 
proof (rule  $\equiv I$ ; rule CP)
  assume  $[(x^P) =_E (y^P) \text{ in } v]$ 
  hence  $[\Box((x^P) =_E (y^P)) \text{ in } v]$ 
    using id-nec3-1[equiv-lr] by auto
  thus  $[\mathcal{A}((x^P) =_E (y^P)) \text{ in } v]$ 

```

```

      using nec-imp-act[deduction] by fast
    next
      assume [ $\mathcal{A}((x^P) =_E (y^P))$  in  $v$ ]
      hence [ $\mathcal{A}(\langle O!, x^P \rangle \ \& \ \langle O!, y^P \rangle \ \& \ \Box(\forall F . \langle F, x^P \rangle \equiv \langle F, y^P \rangle))$  in
 $v$ ]
        apply cut-tac
        apply (PLM-subst-method
          ( $(x^P) =_E (y^P)$ 
            ( $\langle O!, x^P \rangle \ \& \ \langle O!, y^P \rangle \ \& \ \Box(\forall F . \langle F, x^P \rangle \equiv \langle F, y^P \rangle)$ )))
        using eq-E-simple-1 by auto
      hence [ $\mathcal{A}(\langle O!, x^P \rangle \ \& \ \mathcal{A}(\langle O!, y^P \rangle \ \& \ \mathcal{A}(\Box(\forall F . \langle F, x^P \rangle \equiv \langle F, y^P \rangle)))$ 
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]:
    [(( $x^P \neq_E (y^P)$ )  $\equiv$  ( $\mathcal{A}((x^P \neq_E (y^P))$ )) 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: [ $\mathcal{A}(\alpha = \beta)$  in  $v$ ]  $\implies$  [ $(\alpha = \beta)$  in  $v$ ]

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

```

```

    hence  $[(\lambda A! . x^P) \& (\lambda A! . y^P) \& (\Box(\forall F . \lambda x^P . F \equiv \lambda y^P . F))] \text{ in } v]$ 
    using oa-facts-8[equiv-rl] qml-act-2[axiom-instance, equiv-rl]
    &I &E by meson
    hence  $[(x^P) = (y^P) \text{ in } v]$ 
    unfolding identity-defs using  $\forall I$  by auto
  }
  ultimately have  $[(x^P) = (y^P) \text{ in } v]$ 
  using intro-elim-4-a CP by meson
  thus  $[x = y \text{ in } v]$ 
  unfolding identity-defs by auto
qed
end

```

```

instantiation  $\Pi_1 :: id\text{-}act$ 
begin
  instance proof
    interpret PLM .
    fix  $F :: \Pi_1$  and  $G :: \Pi_1$  and  $v :: i$ 
    show  $[\mathcal{A}(F = G) \text{ in } v] \implies [(F = G) \text{ in } v]$ 
    unfolding identity-defs
    using qml-act-2[axiom-instance, equiv-rl] by auto
  qed
end

```

```

instantiation  $\circ :: id\text{-}act$ 
begin
  instance proof
    interpret PLM .
    fix  $p :: \circ$  and  $q :: \circ$  and  $v :: i$ 
    show  $[\mathcal{A}(p = q) \text{ in } v] \implies [p = q \text{ in } v]$ 
    unfolding identityo-def using id-act-prop by blast
  qed
end

```

```

instantiation  $\Pi_2 :: id\text{-}act$ 
begin
  instance proof
    interpret PLM .
    fix  $F :: \Pi_2$  and  $G :: \Pi_2$  and  $v :: i$ 
    assume  $a: [\mathcal{A}(F = G) \text{ in } v]$ 
    {
      fix  $x$ 
      have  $[\mathcal{A}((\lambda y . (\lambda F, x^P . y^P)) = (\lambda y . (\lambda G, x^P . y^P)))$ 
        &  $(\lambda y . (\lambda F, y^P . x^P)) = (\lambda y . (\lambda G, y^P . x^P))) \text{ in } v]$ 
      using a logic-actual-nec-3[axiom-instance, equiv-lr] cqt-basic-4[equiv-lr]
      unfolding identity2-def by blast
      hence  $[((\lambda y . (\lambda F, x^P . y^P)) = (\lambda y . (\lambda G, x^P . y^P)))$ 
        &  $((\lambda y . (\lambda F, y^P . x^P)) = (\lambda y . (\lambda G, y^P . x^P))) \text{ in } v]$ 
      using &I &E id-act-prop Act-Basic-2[equiv-lr] by metis
    }
    thus  $[F = G \text{ in } v]$  unfolding identity-defs by (rule  $\forall I$ )
  qed
end

```

```

instantiation  $\Pi_3 :: id-act$ 
begin
  instance proof
    interpret  $PLM$  .
    fix  $F::\Pi_3$  and  $G::\Pi_3$  and  $v::i$ 
    assume  $a: [\mathcal{A}(F = G) \text{ in } v]$ 
    let  $?p = \lambda x y . (\lambda z. \langle F, z^P, x^P, y^P \rangle) = (\lambda z. \langle G, z^P, x^P, y^P \rangle)$ 
       $\& (\lambda z. \langle F, x^P, z^P, y^P \rangle) = (\lambda z. \langle G, x^P, z^P, y^P \rangle)$ 
       $\& (\lambda z. \langle F, x^P, y^P, z^P \rangle) = (\lambda z. \langle G, x^P, y^P, z^P \rangle)$ 

    {
      fix  $x$ 
      {
        fix  $y$ 
        have  $[\mathcal{A} (?p \ x \ y) \text{ in } v]$ 
        using  $a \text{ logic-actual-nec-3} [axiom-instance, equiv-lr] \text{ cqt-basic-4} [equiv-lr]$ 
        unfolding  $identity_3-def$  by  $blast$ 
        hence  $[?p \ x \ y \text{ in } v]$ 
        using  $\&I \ \&E \ id-act-prop \ Act-Basic-2 [equiv-lr]$  by  $metis$ 
      }
      hence  $[\forall y . ?p \ x \ y \text{ in } v]$ 
      by  $(rule \ \forall I)$ 
    }
    thus  $[F = G \text{ in } v]$ 
    unfolding  $identity_3-def$  by  $(rule \ \forall I)$ 
  qed
end

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

  lemma  $id-act-4[PLM]$ :
     $[((\alpha::('a::id-act)) \neq \beta) \equiv \mathcal{A}(\alpha \neq \beta) \text{ in } v]$ 
    using  $id-act-3 [THEN \ oth-class-taut-5-d [equiv-lr]]$ 
     $logic-actual-nec-1 [axiom-instance, equiv-sym]$ 
     $intro-elim-6-e$  by  $blast$ 

  lemma  $id-act-desc[PLM]$ :
     $[(y^P) = (\iota x . x = y) \text{ in } v]$ 
    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  $eta-conversion-lemma-1[PLM]$ :
     $[(\lambda x . \langle F, x^P \rangle) = F \text{ in } v]$ 
    using  $lambda-predicates-3-1 [axiom-instance]$  .

```


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

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

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

lemma *lambda-p-q-p-eq-q*[PLM]:
 $[(\lambda^0 p) = (\lambda^0 q) \equiv (p = q) \text{ in } v]$
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]:
 $[\forall x . \langle O!, x^P \rangle \vee \langle A!, x^P \rangle \text{ in } v]$
proof (*rule* $\forall I$)
fix *x*
have $[\Diamond \langle E!, x^P \rangle \vee \neg \Diamond \langle E!, x^P \rangle \text{ in } v]$
by *PLM-solver*
moreover have $[\Diamond \langle E!, x^P \rangle \equiv \langle \lambda y . \Diamond \langle E!, y^P \rangle, x^P \rangle \text{ in } v]$
by (*rule* *beta-C-meta-1*[*equiv-sym*]; (*rule* *IsPropositional-intros*))+
moreover have $[\neg \Diamond \langle E!, x^P \rangle \equiv \langle \lambda y . \neg \Diamond \langle E!, y^P \rangle, x^P \rangle \text{ in } v]$
by (*rule* *beta-C-meta-1*[*equiv-sym*]; (*rule* *IsPropositional-intros*))+
ultimately show $[\langle O!, x^P \rangle \vee \langle A!, x^P \rangle \text{ in } v]$
unfolding *Ordinary-def Abstract-def* **by** *PLM-solver*
qed

lemma *partition-2*[PLM]:
 $[\neg(\exists x . \langle O!, x^P \rangle \ \& \ \langle A!, x^P \rangle) \text{ in } v]$
proof –
{
assume $[\exists x . \langle O!, x^P \rangle \ \& \ \langle A!, x^P \rangle \text{ in } v]$
then obtain *b* **where** $[\langle O!, b^P \rangle \ \& \ \langle A!, b^P \rangle \text{ in } v]$
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]:
 $[\langle O!, x \rangle \rightarrow (x =_E x) \text{ in } v]$
proof (*rule* *CP*)

assume $[(\!|O!,x|\!) \text{ in } v]$
moreover have $[\Box(\forall F . (\!|F,x|\!) \equiv (\!|F,x|\!)) \text{ in } v]$
by *PLM-solver*
ultimately show $[(x) =_E (x) \text{ in } v]$
using *&I eq-E-simple-1[equiv-rl]* **by** *blast*
qed

lemma *ord-eq-Eequiv-2[PLM]*:
 $[(x =_E y) \rightarrow (y =_E x) \text{ in } v]$
proof (*rule CP*)
assume $[x =_E y \text{ in } v]$
hence $1: [(\!|O!,x|\!) \ \& \ (\!|O!,y|\!) \ \& \ \Box(\forall F . (\!|F,x|\!) \equiv (\!|F,y|\!)) \text{ in } v]$
using *eq-E-simple-1[equiv-rl]* **by** *simp*
have $[\Box(\forall F . (\!|F,y|\!) \equiv (\!|F,x|\!)) \text{ in } v]$
apply (*PLM-subst1-method*
 $\lambda F . (\!|F,x|\!) \equiv (\!|F,y|\!)$
 $\lambda F . (\!|F,y|\!) \equiv (\!|F,x|\!)$)
using *oth-class-taut-3-g 1[conj2]* **by** *auto*
thus $[y =_E x \text{ in } v]$
using *eq-E-simple-1[equiv-rl] 1[conj1]*
&E &I by *meson*
qed

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

```

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

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

```

```

    apply (rule beta-C-meta-1[equiv-rl])
    unfolding identityE-infix-def
    apply (rule IsPropositional-intros)
    using ord-eq-Eequiv-1[deduction] ord-xy[conj1]
    unfolding identityE-infix-def by presburger
    ultimately have  $[(\lambda z . z^P =_E y^P, x^P) \text{ in } v]$ 
    using l-identity[axiom-instance, deduction, deduction] by fast
    hence  $[x^P =_E y^P \text{ in } v]$ 
    using beta-C-meta-1[equiv-lr] IsPropositional-intros
    unfolding identityE-infix-def by fast
  }
  ultimately show  $[(\lambda z . z^P =_E x^P) \neq (\lambda z . z^P =_E y^P) \text{ in } v]$ 
    using modus-tollens-1 CP by blast
next
  assume ord-xy:  $[(O!, x^P) \ \&\ (\!O!, y^P) \text{ in } v]$ 
  assume  $[(\lambda z . z^P =_E x^P) \neq (\lambda z . z^P =_E y^P) \text{ in } v]$ 
  moreover {
    assume  $[x^P = y^P \text{ in } v]$ 
    hence  $[(\lambda z . z^P =_E x^P) = (\lambda z . z^P =_E y^P) \text{ in } v]$ 
      using id-eq-1 l-identity[axiom-instance, deduction, deduction]
      by fast
  }
  ultimately show  $[x^P \neq y^P \text{ in } v]$ 
    using modus-tollens-1 CP by blast
qed

```

lemma *ab-obey-1[PLM]*:

$[(\lambda A!.x^P) \ \&\ (\lambda A!.y^P)] \rightarrow ((\forall F . \llbracket x^P, F \rrbracket \equiv \llbracket y^P, F \rrbracket) \rightarrow x^P = y^P)$
in v

proof(*rule CP; rule CP*)

assume *abs-xy*: $[(\lambda A!.x^P) \ \&\ (\lambda A!.y^P)] \text{ in } v]$

assume *enc-equiv*: $[\forall F . \llbracket x^P, F \rrbracket \equiv \llbracket y^P, F \rrbracket \text{ in } v]$

{

fix *P*

have $[\llbracket x^P, P \rrbracket \equiv \llbracket y^P, P \rrbracket \text{ in } v]$

using *enc-equiv* by (*rule $\forall E$*)

hence $[\Box(\llbracket x^P, P \rrbracket \equiv \llbracket y^P, P \rrbracket) \text{ in } v]$

using *en-eq-2 intro-elim-6-e intro-elim-6-f*
en-eq-5[equiv-rl] by meson

}

hence $[\Box(\forall F . \llbracket x^P, F \rrbracket \equiv \llbracket y^P, F \rrbracket) \text{ in } v]$

using *BF[deduction] $\forall I$ by fast*

thus $[x^P = y^P \text{ in } v]$

unfolding *identity-defs*

using $\forall I(2)$ *abs-xy &I by presburger*

qed

lemma *ab-obey-2[PLM]*:

$[(\lambda A!.x^P) \ \&\ (\lambda A!.y^P)] \rightarrow ((\exists F . \llbracket x^P, F \rrbracket \ \&\ \neg \llbracket y^P, F \rrbracket) \rightarrow x^P \neq y^P)$
in v

proof(*rule CP; rule CP*)

assume *abs-xy*: $[(\lambda A!.x^P) \ \&\ (\lambda A!.y^P)] \text{ in } v]$

assume $[\exists F . \llbracket x^P, F \rrbracket \ \&\ \neg \llbracket y^P, F \rrbracket \text{ in } v]$

then obtain *P* where *P-prop*:

```

[ $\langle x^P, P \rangle \ \& \ \neg \langle y^P, P \rangle$  in  $v$ ]
by (rule  $\exists E$ )
{
  assume [ $x^P = y^P$  in  $v$ ]
  hence [ $\langle x^P, P \rangle \equiv \langle y^P, P \rangle$  in  $v$ ]
    using l-identity[axiom-instance, deduction, deduction]
    oth-class-taut-4-a by fast
  hence [ $\langle y^P, P \rangle$  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]:

```

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

```

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

```

[ $\Diamond(\exists x . \langle E!, x^P \rangle)$  in  $v$ ]
proof -
  have [ $\Diamond(\exists x . \langle E!, x^P \rangle \ \& \ \Diamond(\neg \langle E!, x^P \rangle))$  in  $v$ ]
    using qml-4[axiom-instance, conj1] .
  hence [ $\Diamond((\exists x . \langle E!, x^P \rangle) \ \& \ (\exists x . \Diamond(\neg \langle E!, x^P \rangle)))$  in  $v$ ]
    using sign-S5-thm-3[deduction] by fast
  hence [ $\Diamond(\exists x . \langle E!, x^P \rangle) \ \& \ \Diamond(\exists x . \Diamond(\neg \langle E!, x^P \rangle))$  in  $v$ ]
    using KBasic2-8[deduction] by blast
  thus ?thesis using &E by blast
qed

```

lemma o-objects-exist-2[PLM]:

```

[ $\Box(\exists x . \langle O!, x^P \rangle)$  in  $v$ ]
apply (rule RN) unfolding Ordinary-def
apply (PLM-subst1-method  $\lambda x . \Diamond \langle E!, x^P \rangle \ \lambda x . (\lambda y . \Diamond \langle E!, y^P \rangle,$ 
 $x^P \rangle)$ 
  apply (rule beta-C-meta-1[equiv-sym], rule IsPropositional-intros)
  using o-objects-exist-1 BF $\Diamond$ [deduction] by blast

```

lemma o-objects-exist-3[PLM]:

```

[ $\Box(\neg(\forall x . \langle A!, x^P \rangle))$  in  $v$ ]
apply (PLM-subst-method  $(\exists x . \neg \langle A!, x^P \rangle) \neg(\forall x . \langle A!, x^P \rangle)$ )
  using cqt-further-2[equiv-sym] apply fast
apply (PLM-subst1-method  $\lambda x . \langle O!, x^P \rangle \ \lambda x . \neg \langle A!, x^P \rangle$ )
  using oa-contingent-2 o-objects-exist-2 by auto

```

lemma a-objects-exist-1[PLM]:

```

[□(∃ x . ⟨A!, xP⟩) in v]
proof –
{
  fix v
  have [∃ x . ⟨A!, xP⟩ & (∀ F . ⟨xP, F⟩ ≡ (F = F)) in v]
    using A-objects[axiom-instance] by simp
  hence [∃ x . ⟨A!, xP⟩ in v]
    using cqt-further-5[deduction, conj1] by fast
}
thus ?thesis by (rule RN)
qed

```

```

lemma a-objects-exist-2[PLM]:
[□(¬(∀ x . ⟨O!, xP⟩)) in v]
apply (PLM-subst-method (∃ x. ¬⟨O!, xP⟩) ¬(∀ x. ⟨O!, xP⟩))
  using cqt-further-2[equiv-sym] apply fast
apply (PLM-subst1-method λ x . ⟨A!, xP⟩ λ x . ¬⟨O!, xP⟩)
  using oa-contingent-3 a-objects-exist-1 by auto

```

```

lemma a-objects-exist-3[PLM]:
[□(¬(∀ x . ⟨E!, xP⟩)) in v]
proof –
{
  fix v
  have [∃ x . ⟨A!, xP⟩ & (∀ F . ⟨xP, F⟩ ≡ (F = F)) in v]
    using A-objects[axiom-instance] by simp
  hence [∃ x . ⟨A!, xP⟩ in v]
    using cqt-further-5[deduction, conj1] by fast
  then obtain a where
    [⟨A!, aP⟩ in v]
    by (rule ∃ E)
  hence [¬(◇⟨E!, aP⟩) in v]
    unfolding Abstract-def
    using beta-C-meta-1[equiv-lr] IsPropositional-intros
    by fast
  hence [¬⟨E!, aP⟩ in v]
    using KBasic2-4[equiv-rl] qml-2[axiom-instance, deduction]
    by simp
  hence [¬(∀ x . ⟨E!, xP⟩) in v]
    using ∃ I cqt-further-2[equiv-rl]
    by fast
}
thus ?thesis
  by (rule RN)
qed

```

```

lemma encoders-are-abstract[PLM]:
[(∃ F . ⟨xP, F⟩) → ⟨A!, xP⟩ 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]:

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

lemma  $obj\text{-}oth\text{-}5[PLM]$ :
  [ $\exists ! x . \langle A!, x^P \rangle \ \& \ (\forall F . \langle x^P, F \rangle \equiv (F = G))$  in  $v$ ]
  using  $A\text{-objects-unique}$  .

```

```

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

```

lemma *A-Exists-1*[PLM]:

$[\mathcal{A}(\exists! x :: ('a :: id-act) . \varphi x) \equiv (\exists! x . \mathcal{A}(\varphi x)) \text{ in } v]$

unfolding *exists-unique-def*

proof (*rule* $\equiv I$; *rule* *CP*)

assume $[\mathcal{A}(\exists \alpha . \varphi \alpha \ \& \ (\forall \beta . \varphi \beta \rightarrow \beta = \alpha)) \text{ in } v]$

hence $[\exists \alpha . \mathcal{A}(\varphi \alpha \ \& \ (\forall \beta . \varphi \beta \rightarrow \beta = \alpha)) \text{ in } v]$

using *Act-Basic-11*[*equiv-lr*] **by** *blast*

then obtain α **where**

$[\mathcal{A}(\varphi \alpha \ \& \ (\forall \beta . \varphi \beta \rightarrow \beta = \alpha)) \text{ in } v]$

by (*rule* $\exists E$)

hence 1: $[\mathcal{A}(\varphi \alpha) \ \& \ \mathcal{A}(\forall \beta . \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

using *Act-Basic-2*[*equiv-lr*] **by** *blast*

find-theorems $\mathcal{A}(?p = ?q)$

have 2: $[\forall \beta . \mathcal{A}(\varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

using 1[*conj2*] *logic-actual-nec-3*[*axiom-instance*, *equiv-lr*] **by** *blast*

{

fix β

have $[\mathcal{A}(\varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

using 2 **by** (*rule* $\forall E$)

hence $[\mathcal{A}(\varphi \beta) \rightarrow (\beta = \alpha) \text{ in } v]$

using *logic-actual-nec-2*[*axiom-instance*, *equiv-lr*, *deduction*]
id-act-3[*equiv-rl*] *CP* **by** *blast*

}

hence $[\forall \beta . \mathcal{A}(\varphi \beta) \rightarrow (\beta = \alpha) \text{ in } v]$

by (*rule* $\forall I$)

thus $[\exists \alpha . \mathcal{A} \varphi \alpha \ \& \ (\forall \beta . \mathcal{A} \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

using 1[*conj1*] $\&I \ \exists I$ **by** *fast*

next

assume $[\exists \alpha . \mathcal{A} \varphi \alpha \ \& \ (\forall \beta . \mathcal{A} \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

then obtain α **where** 1:

$[\mathcal{A} \varphi \alpha \ \& \ (\forall \beta . \mathcal{A} \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

by (*rule* $\exists E$)

{

fix β

have $[\mathcal{A}(\varphi \beta) \rightarrow \beta = \alpha \text{ in } v]$

using 1[*conj2*] **by** (*rule* $\forall E$)

hence $[\mathcal{A}(\varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

using *logic-actual-nec-2*[*axiom-instance*, *equiv-rl*] *id-act-3*[*equiv-lr*]
vdash-properties-10 *CP* **by** *blast*

}

hence $[\forall \beta . \mathcal{A}(\varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

by (*rule* $\forall I$)

hence $[\mathcal{A}(\forall \beta . \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$

using *logic-actual-nec-3*[*axiom-instance*, *equiv-rl*] **by** *fast*

hence $[\mathcal{A}(\varphi \alpha \ \& \ (\forall \beta . \varphi \beta \rightarrow \beta = \alpha)) \text{ in } v]$

using 1[*conj1*] *Act-Basic-2*[*equiv-rl*] $\&I$ **by** *blast*

hence $[\exists \alpha . \mathcal{A}(\varphi \alpha \ \& \ (\forall \beta . \varphi \beta \rightarrow \beta = \alpha)) \text{ in } v]$

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

thus $[\mathcal{A}(\exists \alpha . \varphi \alpha \ \& \ (\forall \beta . \varphi \beta \rightarrow \beta = \alpha)) \text{ in } v]$

using *Act-Basic-11*[*equiv-rl*] **by** *fast*

qed

lemma *A-Exists-2*[PLM]:

$[(\exists y . y^P = (\iota x . \varphi x)) \equiv \mathcal{A}(\exists! x . \varphi x) \text{ in } v]$

using *actual-desc-1 A-Exists-1*[*equiv-sym*]
intro-elim-6-e **by** *blast*

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

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

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

lemma *desc-encode*[*PLM*]:
 $[(\iota x . \langle A!, x^P \rangle) \ \& \ (\forall F . \langle x^P, F \rangle \equiv \varphi F), G \equiv \varphi G \text{ in } dw]$
proof –
obtain *a* **where**
 $[a^P = (\iota x . \langle A!, x^P \rangle) \ \& \ (\forall F . \langle x^P, F \rangle \equiv \varphi F) \text{ in } dw]$
using *A-descriptions* **by** (*rule* $\exists E$)
moreover **hence** $[\langle a^P, G \rangle \equiv \varphi G \text{ in } dw]$
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 $?x = \iota x . \langle A!, x^P \rangle \ \& \ (\forall F . \langle x^P, F \rangle \equiv (\exists q . q \ \& \ F = (\lambda y . q)))$
have $[(\exists p . \text{ContingentlyTrue } p) \text{ in } dw]$
using *cont-tf-thm-3* **by** *auto*
then **obtain** p_1 **where** $[\text{ContingentlyTrue } p_1 \text{ in } dw]$ **by** (*rule* $\exists E$)
hence $[p_1 \text{ in } dw]$ **unfolding** *ContingentlyTrue-def* **using** **&E** **by** *fast*
hence $[p_1 \ \& \ (\lambda y . p_1) = (\lambda y . p_1) \text{ in } dw]$ **using** *&I id-eq-1* **by** *fast*
hence $[\exists q . q \ \& \ (\lambda y . p_1) = (\lambda y . q) \text{ in } dw]$ **using** $\exists I$ **by** *fast*
moreover **have** $[\langle ?x, \lambda y . p_1 \rangle \equiv (\exists q . q \ \& \ (\lambda y . p_1) = (\lambda y . q)) \text{ in } dw]$
using *desc-encode* **by** *fast*
ultimately **have** $[\langle ?x, \lambda y . p_1 \rangle \text{ in } dw]$
using $\equiv E$ **by** *blast*
hence $[\Box \langle ?x, \lambda y . p_1 \rangle \text{ in } dw]$
using *encoding*[*axiom-instance, deduction*] **by** *fast*

```

    hence  $\forall v . [\llbracket ?x, \lambda y . p_1 \rrbracket \text{ in } v]$ 
    using Semantics.T6 by simp
end

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

notepad
begin
  fix v
  let  $?x = \iota x . (\llbracket A!, x^P \rrbracket) \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv (\exists q . q \ \& \ F = (\lambda y .$ 
   $q)))$ 
  have  $[\Box(\exists p . \text{ContingentlyTrue } p) \text{ in } v]$ 
    using cont-tf-thm-3 RN by auto
  hence  $[\mathcal{A}(\exists p . \text{ContingentlyTrue } p) \text{ in } v]$ 
    using nec-imp-act[deduction] by simp
  hence  $[\exists p . \mathcal{A}(\text{ContingentlyTrue } p) \text{ in } v]$ 
    using Act-Basic-11[equiv-lr] by auto
  then obtain  $p_1$  where
     $[\mathcal{A}(\text{ContingentlyTrue } p_1) \text{ in } v]$ 
    by (rule  $\exists E$ )
  hence  $[\mathcal{A}p_1 \text{ in } v]$ 
    unfolding ContingentlyTrue-def
    using Act-Basic-2[equiv-lr]  $\&E$  by fast
  hence  $[\mathcal{A}p_1 \ \& \ \mathcal{A}((\lambda y . p_1) = (\lambda y . p_1)) \text{ in } v]$ 
    using  $\&I$  id-eq-1[THEN RN, THEN nec-imp-act[deduction]] by
fast
  hence  $[\mathcal{A}(p_1 \ \& \ (\lambda y . p_1) = (\lambda y . p_1)) \text{ in } v]$ 
    using Act-Basic-2[equiv-rl] by fast
  hence  $[\exists q . \mathcal{A}(q \ \& \ (\lambda y . p_1) = (\lambda y . q)) \text{ in } v]$ 
    using  $\exists I$  by fast
  hence  $[\mathcal{A}(\exists q . q \ \& \ (\lambda y . p_1) = (\lambda y . q)) \text{ in } v]$ 
    using Act-Basic-11[equiv-rl] by fast

```

moreover have $\llbracket ?x, \lambda y . p_1 \rrbracket \equiv \mathcal{A}(\exists q . q \ \& \ (\lambda y . p_1) = (\lambda y . q))$ in v
 using *desc-nec-encode* by *fast*
 ultimately have $\llbracket ?x, \lambda y . p_1 \rrbracket$ in v
 using $\equiv E$ by *blast*
 end

lemma *Box-desc-encode-1*[PLM]:

$\llbracket \Box(\varphi \ G) \rightarrow \llbracket (\lambda x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)), G \rrbracket$ in v

proof (rule *CP*)

assume $\llbracket \Box(\varphi \ G) \rrbracket$ in v

hence $\llbracket \mathcal{A}(\varphi \ G) \rrbracket$ in v

using *nec-imp-act*[*deduction*] by *auto*

thus $\llbracket (\lambda x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)), G \rrbracket$ in v

using *desc-nec-encode*[*equiv-rl*] by *simp*

qed

lemma *Box-desc-encode-2*[PLM]:

$\llbracket \Box(\varphi \ G) \rightarrow \Box(\llbracket (\lambda x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)), G \rrbracket \equiv \varphi \ G) \rrbracket$ in v

proof (rule *CP*)

assume a : $\llbracket \Box(\varphi \ G) \rrbracket$ in v

hence $\llbracket \Box(\llbracket (\lambda x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)), G \rrbracket \rightarrow \varphi \ G) \rrbracket$

in v

using *KBasic-1*[*deduction*] by *simp*

moreover {

have $\llbracket \llbracket (\lambda x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)), G \rrbracket \rrbracket$ in v

using *a Box-desc-encode-1*[*deduction*] by *auto*

hence $\llbracket \Box(\llbracket (\lambda x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)), G \rrbracket \rrbracket$ in v

using *encoding*[*axiom-instance, deduction*] by *blast*

hence $\llbracket \Box(\varphi \ G \rightarrow \llbracket (\lambda x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)), G \rrbracket \rrbracket$ in v

$G \rrbracket$ in v

using *KBasic-1*[*deduction*] by *simp*

}

ultimately show $\llbracket \Box(\llbracket (\lambda x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)), G \rrbracket$

$G \rrbracket$

$\equiv \varphi \ G) \rrbracket$ in v

using $\&I$ *KBasic-4*[*equiv-rl*] by *blast*

qed

lemma *box-phi-a-1*[PLM]:

assumes $\llbracket \Box(\forall F . \varphi \ F \rightarrow \Box(\varphi \ F)) \rrbracket$ in v

shows $\llbracket (\llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)) \rightarrow \Box(\llbracket A!, x^P \rrbracket$

$\ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)) \rrbracket$ in v

proof (rule *CP*)

assume a : $\llbracket (\llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F)) \rrbracket$ in v

have $\llbracket \Box(\llbracket A!, x^P \rrbracket) \rrbracket$ in v

using *oa-facts-2*[*deduction*] *a[conj1]* by *auto*

moreover have $\llbracket \Box(\forall F . \llbracket x^P, F \rrbracket \equiv \varphi \ F) \rrbracket$ in v

proof (rule *BF*[*deduction*]; rule $\forall I$)

fix F

have ϑ : $\llbracket \Box(\varphi \ F \rightarrow \Box(\varphi \ F)) \rrbracket$ in v

using *assms*[*THEN CBF*[*deduction*]] by (rule $\forall E$)

moreover have $\llbracket \Box(\llbracket x^P, F \rrbracket \rightarrow \Box(\llbracket x^P, F \rrbracket)) \rrbracket$ in v

```

    using encoding[axiom-necessitation, axiom-instance] by simp
  moreover have  $[\Box \langle x^P, F \rangle \equiv \Box(\varphi F) \text{ in } v]$ 
  proof (rule  $\equiv I$ ; rule CP)
    assume  $[\Box \langle x^P, F \rangle \text{ in } v]$ 
    hence  $[\langle x^P, F \rangle \text{ in } v]$ 
    using qml-2[axiom-instance, deduction] by blast
    hence  $[\varphi F \text{ in } v]$ 
    using a[conj2]  $\forall E \equiv E$  by blast
    thus  $[\Box(\varphi F) \text{ in } v]$ 
    using  $\vartheta[THEN \text{ qml-2}[axiom-instance, deduction], deduction]$ 
  by simp
  next
    assume  $[\Box(\varphi F) \text{ in } v]$ 
    hence  $[\varphi F \text{ in } v]$ 
    using qml-2[axiom-instance, deduction] by blast
    hence  $[\langle x^P, F \rangle \text{ in } v]$ 
    using a[conj2]  $\forall E \equiv E$  by blast
    thus  $[\Box \langle x^P, F \rangle \text{ in } v]$ 
    using encoding[axiom-instance, deduction] by simp
  qed
  ultimately show  $[\Box(\langle x^P, F \rangle \equiv \varphi F) \text{ in } v]$ 
  using sc-eq-box-box-3[deduction, deduction] &I by blast
  qed
  ultimately show  $[\Box(\langle A!, x^P \rangle \ \& \ (\forall F. \langle x^P, F \rangle \equiv \varphi F)) \text{ in } v]$ 
  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  $[\Box(\forall F. \varphi F \rightarrow \Box(\varphi F)) \text{ in } v]$ 
  shows  $[y^P = (\iota x. \langle A!, x^P \rangle) \ \& \ (\forall F. \langle x^P, F \rangle \equiv \varphi F) \rightarrow (\langle A!, y^P \rangle \ \& \ (\forall F. \langle y^P, F \rangle \equiv \varphi F)) \text{ in } v]$ 
  proof -
    let  $?\psi = \lambda x. \langle A!, x^P \rangle \ \& \ (\forall F. \langle x^P, F \rangle \equiv \varphi F)$ 
    have  $[\forall x. ?\psi x \rightarrow \Box(?\psi x) \text{ in } v]$ 
    using box-phi-a-1[OF assms]  $\forall I$  by fast
    hence  $[(\exists! x. ?\psi x) \rightarrow (\forall y. y^P = (\iota x. ?\psi x) \rightarrow ?\psi y) \text{ in } v]$ 
    using unique-box-desc[deduction] by fast
    hence  $[(\forall y. y^P = (\iota x. ?\psi x) \rightarrow ?\psi y) \text{ in } v]$ 
    using A-objects-unique modus-ponens by blast
    thus ?thesis by (rule  $\forall E$ )
  qed

```

```

lemma box-phi-a-3[PLM]:
  assumes  $[\Box(\forall F. \varphi F \rightarrow \Box(\varphi F)) \text{ in } v]$ 
  shows  $[(\iota x. \langle A!, x^P \rangle) \ \& \ (\forall F. \langle x^P, F \rangle \equiv \varphi F), G \equiv \varphi G \text{ in } v]$ 
  proof -
    obtain a where
       $[a^P = (\iota x. \langle A!, x^P \rangle) \ \& \ (\forall F. \langle x^P, F \rangle \equiv \varphi F) \text{ in } v]$ 
      using A-descriptions by (rule  $\exists E$ )
    moreover {
      hence  $[(\forall F. \langle a^P, F \rangle \equiv \varphi F) \text{ in } v]$ 

```

```

    using box-phi-a-2[OF assms, deduction, conj2] by blast
  hence  $\llbracket a^P, G \rrbracket \equiv \varphi \ G \text{ in } v$  by (rule  $\forall E$ )
}
ultimately show ?thesis
  using l-identity[axiom-instance, deduction, deduction] by fast
qed

```

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

$[\exists! x . \text{Null } (x^P) \text{ in } v]$

proof –

have $[\exists x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv (F \neq F)) \text{ in } v]$

using *A-objects*[axiom-instance] by simp

then obtain *a* **where** *a-prop*:

$[\llbracket A!, a^P \rrbracket \ \& \ (\forall F . \llbracket a^P, F \rrbracket \equiv (F \neq F)) \text{ in } v]$

by (rule $\exists E$)

have *1*: $[\llbracket A!, a^P \rrbracket \ \& \ (\neg(\exists F . \llbracket a^P, F \rrbracket)) \text{ in } v]$

using *a-prop*[conj1] **apply** (rule $\&I$)

proof –

{

assume $[\exists F . \llbracket a^P, F \rrbracket \text{ in } v]$

then obtain *P* **where**

$[\llbracket a^P, P \rrbracket \text{ in } v]$ **by** (rule $\exists E$)

hence $[P \neq P \text{ in } v]$

using *a-prop*[conj2, THEN $\forall E$, equiv-lr] **by** simp

hence $[\neg(\exists F . \llbracket a^P, F \rrbracket) \text{ in } v]$

using *id-eq-1* *reductio-aa-1* **by** fast

}

thus $[\neg(\exists F . \llbracket a^P, F \rrbracket) \text{ in } v]$

using *reductio-aa-1* **by** blast

qed

moreover have $[\forall y . (\llbracket A!, y^P \rrbracket \ \& \ (\neg(\exists F . \llbracket y^P, F \rrbracket))) \rightarrow y = a$

in v]

proof (rule $\forall I$; rule *CP*)

fix *y*

assume *2*: $[\llbracket A!, y^P \rrbracket \ \& \ (\neg(\exists F . \llbracket y^P, F \rrbracket)) \text{ in } v]$

have $[\forall F . \llbracket y^P, F \rrbracket \equiv \llbracket a^P, F \rrbracket \text{ in } v]$

using *cqt-further-12*[deduction] *1*[conj2] *2*[conj2] $\&I$ **by** blast

thus $[y = a \text{ in } v]$

using *ab-obey-1*[deduction, deduction]

$\&I$ [OF *2*[conj1] *1*[conj1]] *identity- ν -def* **by** *presburger*

qed

ultimately show ?thesis

using $\&I \exists I$

unfolding *Null-def exists-unique-def* **by** fast

qed

lemma *null-uni-uniq-2*[PLM]:

$[\exists! x . \text{Universal } (x^P) \text{ in } v]$

proof –

have $[\exists x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv (F = F)) \text{ in } v]$

using *A-objects*[axiom-instance] **by** simp

then obtain *a* **where** *a-prop*:

$[\llbracket A!, a^P \rrbracket \ \& \ (\forall F . \llbracket a^P, F \rrbracket \equiv (F = F)) \text{ in } v]$

by (rule $\exists E$)

```

have 1: [(A!, aP) & (∀ F . ‖aP, 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 [∀ y . ((A!, yP) & (∀ F . ‖yP, F‖)) → y = a] in
v]
proof (rule ∀ I; rule CP)
  fix y
  assume 2: [(A!, yP) & (∀ F . ‖yP, F‖)] in v
  have [∀ F . ‖yP, F‖ ≡ ‖aP, 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]:
  [∃ y . yP = (ιx . Null (xP)) in v]
using null-uni-uniq-1[THEN RN, THEN nec-imp-act[deduction]]
  A-Exists-2[equiv-rl] by auto

lemma null-uni-uniq-4[PLM]:
  [∃ y . yP = (ιx . Universal (xP)) 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 (xP) → □(Null (xP)) in v]
proof (rule CP)
  assume [Null (xP) in v]
  hence 1: [(A!, xP) & (¬(∃ F . ‖xP, F‖))] in v
  unfolding Null-def .
  have [□(A!, xP) in v]
  using 1[conj1] oa-facts-2[deduction] by simp
  moreover have [□(¬(∃ F . ‖xP, F‖))] in v
  proof -
    {
      assume [¬□(¬(∃ F . ‖xP, F‖))] in v
      hence [◇(∃ F . ‖xP, F‖)] in v
      unfolding diamond-def .
      hence [∃ F . ◇‖xP, F‖] in v
      using BF◇[deduction] by blast
      then obtain P where [◇‖xP, P‖] in v
      by (rule ∃ E)
      hence [‖xP, P‖] in v
      using en-eq-3[equiv-lr] by simp
      hence [∃ F . ‖xP, F‖] in v
      using ∃ I by blast
    }
  thus ?thesis

```

```

      using 1[conj2] modus-tollens-1 CP
      useful-tautologies-1[deduction] by metis
    qed
  ultimately show  $\Box \text{Null } (x^P) \text{ in } v$ 
    unfolding Null-def
    using &I KBasic-3[equiv-rl] by blast
  qed

```

```

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

```

```

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

```

```

lemma null-uni-facts-4[PLM]:
   $[ \text{Universal } (\mathbf{a}_V) \text{ in } v ]$ 
proof -
  let  $? \psi = \lambda x . \text{Universal } x$ 

```

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* $\forall I$] **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* $\forall I$] **by** *fast*
have $[\exists y . y^P = (a_V)]$ *in v*
unfolding *UniversalObject-def* **using** *null-uni-uniq-4* .
then obtain y **where** $[y^P = (a_V)]$ *in v*
by (*rule* $\exists E$)
moreover hence $[?\psi(y^P)]$ *in v*
using $1[THEN \forall E, deduction]$
unfolding *UniversalObject-def* **by** *simp*
ultimately show $[?\psi(a_V)]$ *in v*
using *l-identity*[*axiom-instance*, deduction, deduction] **by** *blast*
qed

lemma *aclassical-1*[*PLM*]:

$[\forall R . \exists x y . (A!, x^P) \ \& \ (A!, y^P) \ \& \ (x \neq y)$
 $\ \& \ (\lambda z . (R, z^P, x^P)) = (\lambda z . (R, z^P, y^P))]$ *in v*

proof (*rule* $\forall I$)

fix *R*

obtain a **where** ϑ :

$[(A!, a^P) \ \& \ (\forall F . \llbracket a^P, F \rrbracket \equiv (\exists y . (A!, y^P)$
 $\ \& \ F = (\lambda z . (R, z^P, y^P)) \ \& \ \neg \llbracket y^P, F \rrbracket))]$ *in v*
using *A-objects*[*axiom-instance*] **by** (*rule* $\exists E$)

{
assume $[\neg \llbracket a^P, (\lambda z . (R, z^P, a^P)) \rrbracket]$ *in v*
hence $[\neg ((A!, a^P) \ \& \ (\lambda z . (R, z^P, a^P)) = (\lambda z . (R, z^P, a^P))$
 $\ \& \ \neg \llbracket a^P, (\lambda z . (R, z^P, a^P)) \rrbracket)]$ *in v*
using $\vartheta[conj2, THEN \forall E, THEN oth-class-taut-5-d[equiv-lr],$
equiv-lr]

cqt-further-4[*equiv-lr*] $\forall E$ **by** *blast*

hence $[(A!, a^P) \ \& \ (\lambda z . (R, z^P, a^P)) = (\lambda z . (R, z^P, a^P))$
 $\rightarrow \llbracket a^P, (\lambda z . (R, z^P, a^P)) \rrbracket]$ *in v*

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

hence $[\llbracket a^P, (\lambda z . (R, z^P, a^P)) \rrbracket]$ *in v*

using $\vartheta[conj1]$ *id-eq-1* $\& I$ *vdash-properties-10* **by** *fast*

}

hence $1: [\llbracket a^P, (\lambda z . (R, z^P, a^P)) \rrbracket]$ *in v*

using *reductio-aa-1* *CP if-p-then-p* **by** *blast*

then obtain b **where** ξ :

$[(A!, b^P) \ \& \ (\lambda z . (R, z^P, a^P)) = (\lambda z . (R, z^P, b^P))$
 $\ \& \ \neg \llbracket b^P, (\lambda z . (R, z^P, a^P)) \rrbracket]$ *in v*

using $\vartheta[conj2, THEN \forall E, equiv-lr]$ $\exists E$ **by** *blast*

have $[a \neq b]$ *in v*

proof –

{

assume $[a = b]$ *in v*

hence $[\llbracket b^P, (\lambda z . (R, z^P, a^P)) \rrbracket]$ *in v*

using 1 *l-identity*[*axiom-instance*, deduction, deduction] **by**

fast

hence *?thesis*


```

    using  $\xi[conj2]$  reductio-aa-1 by blast
  }
  thus ?thesis using reductio-aa-1 by blast
qed
hence  $[(A!, a^P) \ \& \ (A!, b^P) \ \& \ a \neq b$ 
   $\ \& \ (\lambda z. (R, z^P, a^P)) = (\lambda z. (R, z^P, b^P)) \text{ in } v]$ 
  using  $\vartheta[conj1] \ \xi[conj1, conj1] \ \xi[conj1, conj2] \ \&I$  by presburger
hence  $[\exists y. (A!, a^P) \ \& \ (A!, y^P) \ \& \ a \neq y$ 
   $\ \& \ (\lambda z. (R, z^P, a^P)) = (\lambda z. (R, z^P, y^P)) \text{ in } v]$ 
  using  $\exists I$  by fast
thus  $[\exists x y. (A!, x^P) \ \& \ (A!, y^P) \ \& \ x \neq y$ 
   $\ \& \ (\lambda z. (R, z^P, x^P)) = (\lambda z. (R, z^P, y^P)) \text{ in } v]$ 
  using  $\exists I$  by fast
qed

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

```

hence $[(\downarrow A!, a^P) \ \& \ (\downarrow A!, b^P) \ \& \ a \neq b$
 $\ \& \ (\lambda z. (\downarrow R, a^P, z^P)) = (\lambda z. (\downarrow R, b^P, z^P)) \text{ in } v]$
 using $\vartheta[\text{conj1}] \ \xi[\text{conj1}, \text{conj1}] \ \xi[\text{conj1}, \text{conj2}] \ \&I$ by *presburger*
 hence $[\exists y. (\downarrow A!, a^P) \ \& \ (\downarrow A!, y^P) \ \& \ a \neq y$
 $\ \& \ (\lambda z. (\downarrow R, a^P, z^P)) = (\lambda z. (\downarrow R, y^P, z^P)) \text{ in } v]$
 using $\exists I$ by *fast*
 thus $[\exists x y. (\downarrow A!, x^P) \ \& \ (\downarrow A!, y^P) \ \& \ x \neq y$
 $\ \& \ (\lambda z. (\downarrow R, x^P, z^P)) = (\lambda z. (\downarrow R, y^P, z^P)) \text{ in } v]$
 using $\exists I$ by *fast*
 qed

lemma *aclassical-3[PLM]*:

$[\forall F. \exists x y. (\downarrow A!, x^P) \ \& \ (\downarrow A!, y^P) \ \& \ (x \neq y)$
 $\ \& \ ((\lambda^0 (\downarrow F, x^P)) = (\lambda^0 (\downarrow F, y^P))) \text{ in } v]$

proof (rule $\forall I$)

fix R

obtain a where ϑ :

$[(\downarrow A!, a^P) \ \& \ (\forall F. \downarrow a^P, F) \equiv (\exists y. (\downarrow A!, y^P)$
 $\ \& \ F = (\lambda z. (\downarrow R, y^P)) \ \& \ \neg \downarrow y^P, F)) \text{ in } v]$

using *A-objects[axiom-instance]* by (rule $\exists E$)

{

assume $[\neg \downarrow a^P, (\lambda z. (\downarrow R, a^P))] \text{ in } v]$

hence $[\neg((\downarrow A!, a^P) \ \& \ (\lambda z. (\downarrow R, a^P))) = (\lambda z. (\downarrow R, a^P))$

$\ \& \ \neg \downarrow a^P, (\lambda z. (\downarrow R, a^P))] \text{ in } v]$

using $\vartheta[\text{conj2}, \text{THEN } \forall E, \text{THEN } \text{oth-class-taut-5-d}[\text{equiv-lr}],$

equiv-lr]

cqt-further-4[equiv-lr] $\forall E$ by blast

hence $[(\downarrow A!, a^P) \ \& \ (\lambda z. (\downarrow R, a^P)) = (\lambda z. (\downarrow R, a^P))$

$\rightarrow \downarrow a^P, (\lambda z. (\downarrow R, a^P))] \text{ in } v]$

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

hence $[\downarrow a^P, (\lambda z. (\downarrow R, a^P))] \text{ in } v]$

using $\vartheta[\text{conj1}]$ *id-eq-1* $\&I$ *vdash-properties-10* by *fast*

}

hence $1: [\downarrow a^P, (\lambda z. (\downarrow R, a^P))] \text{ in } v]$

using *reductio-aa-1 CP if-p-then-p* by *blast*

then obtain b where ξ :

$[(\downarrow A!, b^P) \ \& \ (\lambda z. (\downarrow R, a^P)) = (\lambda z. (\downarrow R, b^P))$

$\ \& \ \neg \downarrow b^P, (\lambda z. (\downarrow R, a^P))] \text{ in } v]$

using $\vartheta[\text{conj2}, \text{THEN } \forall E, \text{equiv-lr}] \ \exists E$ by *blast*

have $[a \neq b \text{ in } v]$

proof –

{

assume $[a = b \text{ in } v]$

hence $[\downarrow b^P, (\lambda z. (\downarrow R, a^P))] \text{ in } v]$

using *1 l-identity[axiom-instance, deduction, deduction]* by

fast

hence *?thesis*

using $\xi[\text{conj2}]$ *reductio-aa-1* by *blast*

}

thus *?thesis* using *reductio-aa-1* by *blast*

qed

moreover {

have $[(\downarrow R, a^P) = (\downarrow R, b^P) \text{ in } v]$

unfolding *identity_o-def*

```

    using  $\xi[conj1, conj2]$  by auto
    hence  $[(\lambda^0 \langle R, a^P \rangle) = (\lambda^0 \langle R, b^P \rangle)]$  in  $v$ 
    using lambda-p-q-p-eg-q[equiv-rl] by simp
  }
ultimately have  $[(\langle A!, a^P \rangle \ \& \ \langle A!, b^P \rangle \ \& \ a \neq b$ 
  &  $((\lambda^0 \langle R, a^P \rangle) = (\lambda^0 \langle R, b^P \rangle))$  in  $v]$ 
  using  $\vartheta[conj1]$   $\xi[conj1, conj1]$   $\xi[conj1, conj2]$  & I
  by presburger
hence  $[\exists y . (\langle A!, a^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ a \neq y$ 
  &  $(\lambda^0 \langle R, a^P \rangle) = (\lambda^0 \langle R, y^P \rangle)]$  in  $v]$ 
  using  $\exists I$  by fast
thus  $[\exists x y . (\langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ x \neq y$ 
  &  $(\lambda^0 \langle R, x^P \rangle) = (\lambda^0 \langle R, y^P \rangle)]$  in  $v]$ 
  using  $\exists I$  by fast
qed

lemma aclassical2[PLM]:
   $[\exists x y . (\langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ x \neq y \ \& \ (\forall F . \langle F, x^P \rangle \equiv \langle F, y^P \rangle))$ 
  in  $v]$ 
proof -
  let  $?R_1 = \lambda^2 (\lambda x y . \forall F . \langle F, x^P \rangle \equiv \langle F, y^P \rangle)$ 
  have  $[\exists x y . (\langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ x \neq y$ 
    &  $(\lambda z . \langle ?R_1, z^P, x^P \rangle) = (\lambda z . \langle ?R_1, z^P, y^P \rangle)]$  in  $v]$ 
    using aclassical-1 by (rule  $\forall E$ )
  then obtain a where
     $[\exists y . (\langle A!, a^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ a \neq y$ 
      &  $(\lambda z . \langle ?R_1, z^P, a^P \rangle) = (\lambda z . \langle ?R_1, z^P, y^P \rangle)]$  in  $v]$ 
    by (rule  $\exists E$ )
  then obtain b where ab-prop:
     $[(\langle A!, a^P \rangle \ \& \ \langle A!, b^P \rangle \ \& \ a \neq b$ 
      &  $(\lambda z . \langle ?R_1, z^P, a^P \rangle) = (\lambda z . \langle ?R_1, z^P, b^P \rangle)]$  in  $v]$ 
    by (rule  $\exists E$ )
  have  $[(\langle ?R_1, a^P, a^P \rangle)]$  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  $[(\lambda z . \langle ?R_1, z^P, a^P \rangle, a^P)]$  in  $v]$ 
    apply cut-tac apply (rule beta-C-meta-1[equiv-rl])
    apply (rule IsPropositional-intros)
    by auto
  hence  $[(\lambda z . \langle ?R_1, z^P, b^P \rangle, a^P)]$  in  $v]$ 
    using ab-prop[conj2] l-identity[axiom-instance, deduction, deduc-
    tion]
    by fast
  hence  $[(\langle ?R_1, a^P, b^P \rangle)]$  in  $v]$ 
    using beta-C-meta-1[equiv-lr] IsPropositional-intros by fast
  hence  $[\forall F . \langle F, a^P \rangle \equiv \langle F, b^P \rangle]$  in  $v]$ 
    using beta-C-meta-2[equiv-lr] IsPropositional-intros by fast
  hence  $[(\langle A!, a^P \rangle \ \& \ \langle A!, b^P \rangle \ \& \ a \neq b \ \& \ (\forall F . \langle F, a^P \rangle \equiv \langle F, b^P \rangle))$ 
  in  $v]$ 
    using ab-prop[conj1] & I by presburger
  hence  $[\exists y . (\langle A!, a^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ a \neq y \ \& \ (\forall F . \langle F, a^P \rangle \equiv$ 
   $\langle F, y^P \rangle))]$  in  $v]$ 
    using  $\exists I$  by fast

```

```

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

```

9.13 Propositional Properties

```

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

lemma prop-prop2-2:
   $[F = (\lambda x . p) \rightarrow \Box(\forall x . \langle F, x^P \rangle \equiv p) \text{ in } v]$ 
proof (rule CP)
  assume 1:  $[F = (\lambda x . p) \text{ in } v]$ 
  {
    fix v
    {
      fix x
      have  $[\langle (\lambda x . p), x^P \rangle \equiv p \text{ in } v]$ 
        apply (rule beta-C-meta-1)
        by (rule IsPropositional-intros)+
    }
    hence  $[\forall x . \langle (\lambda x . p), x^P \rangle \equiv p \text{ in } v]$ 
      by (rule  $\forall I$ )
  }
  hence  $[\Box(\forall x . \langle (\lambda x . p), x^P \rangle \equiv p) \text{ in } v]$ 
    by (rule RN)
  thus  $[\Box(\forall x . \langle F, x^P \rangle \equiv p) \text{ in } v]$ 
    using l-identity[axiom-instance,deduction,deduction,
      OF 1[THEN id-eq-prop-prop-2[deduction]]] by fast
qed

```

```

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

```

lemma *prop-indis*:

```

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

```

lemma *prop-in-thm*:

```

[Propositional  $F \rightarrow$  Indiscriminate  $F$  in  $v$ ]
proof (rule CP)
  assume [Propositional  $F$  in  $v$ ]
  hence [ $\Box(\text{Propositional } F)$  in  $v$ ]
    using prop-prop2-3[deduction] by auto
  moreover {
    fix  $w$ 
    assume [ $\exists p . (F = (\lambda y . p))$  in  $w$ ]
    then obtain  $q$  where  $q\text{-prop}$ : [ $F = (\lambda y . q)$  in  $w$ ]
      by (rule  $\exists E$ )
    {
      assume [ $\exists x . \langle F, x^P \rangle$  in  $w$ ]
      then obtain  $a$  where [ $\langle F, a^P \rangle$  in  $w$ ]
        by (rule  $\exists E$ )
      hence [ $\langle \lambda y . q, a^P \rangle$  in  $w$ ]
        using  $q\text{-prop } l\text{-identity}$  [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 [ $\langle \lambda y . q, x^P \rangle$  in  $w$ ]

```

```

      using q beta-C-meta-1 [equiv-rl] IsPropositional-intros by fast
    hence  $\llbracket F, x^P \rrbracket$  in w
      using q-prop [eq-sym] l-identity [axiom-instance, deduction,
deduction]
    by fast
  }
  hence  $\llbracket \forall x . \llbracket F, x^P \rrbracket \rrbracket$  in w
    by (rule  $\forall I$ )
  }
  hence  $\llbracket (\exists x . \llbracket F, x^P \rrbracket) \rightarrow (\forall x . \llbracket F, x^P \rrbracket) \rrbracket$  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  $\rightarrow$  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  $\rightarrow$  Indiscriminate F in v]
  proof -
    {
      fix w
      have  $\llbracket (\neg(\exists x . \llbracket F, x^P \rrbracket)) \rightarrow ((\exists x . \llbracket F, x^P \rrbracket) \rightarrow (\forall x . \llbracket F, x^P \rrbracket)) \rrbracket$ 
in w
        using useful-tautologies-3 by auto
      hence  $\llbracket (\forall x . \neg \llbracket F, x^P \rrbracket) \rightarrow ((\exists x . \llbracket F, x^P \rrbracket) \rightarrow (\forall x . \llbracket F, x^P \rrbracket)) \rrbracket$ 
in w
        apply cut-tac apply (PLM-subst-method  $\neg(\exists x . \llbracket F, x^P \rrbracket) (\forall x . \neg \llbracket F, x^P \rrbracket)$ )
        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:
   $\llbracket \neg(\text{Indiscriminate } E!) \rrbracket$  in v
  proof (rule reductio-aa-2)
    show  $\llbracket \Box \neg(\forall x . \llbracket E!, x^P \rrbracket) \rrbracket$  in v
      using a-objects-exist-3 .
    next
      assume [Indiscriminate E! in v]
      thus  $\llbracket \neg \Box \neg(\forall x . \llbracket E!, x^P \rrbracket) \rrbracket$  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:
  [ $\neg(\text{Indiscriminate } (E!^-))$  in  $v$ ]
proof (rule reductio-aa-2)
  assume [ $\text{Indiscriminate } (E!^-)$  in  $v$ ]
  moreover have [ $\Box(\exists x . \langle E!^-, x^P \rangle)$  in  $v$ ]
  apply (PLM-subst1-method  $\lambda x . \neg \langle E!, x^P \rangle \lambda x . \langle E!^-, x^P \rangle$ )
  using thm-relation-negation-1-1 [equiv-sym] apply simp
  unfolding exists-def
  apply (PLM-subst1-method  $\lambda x . \langle E!, x^P \rangle \lambda x . \neg \neg \langle E!, x^P \rangle$ )
  using oth-class-taut-4-b apply simp
  using a-objects-exist-3 by auto
  ultimately have [ $\Box(\forall x . \langle E!^-, x^P \rangle)$  in  $v$ ]
  unfolding Indiscriminate-def
  using qml-1 [axiom-instance, deduction, deduction] by blast
  thus [ $\Box(\forall x . \neg \langle E!, x^P \rangle)$  in  $v$ ]
  apply cut-tac
  apply (PLM-subst1-method  $\lambda x . \langle E!^-, x^P \rangle \lambda x . \neg \langle E!, x^P \rangle$ )
  using thm-relation-negation-1-1 by auto
next
  show [ $\neg \Box(\forall x . \neg \langle E!, x^P \rangle)$  in  $v$ ]
  using o-objects-exist-1
  unfolding diamond-def exists-def
  apply cut-tac
  apply (PLM-subst-method  $\neg \neg(\forall x . \neg \langle E!, x^P \rangle) \forall x . \neg \langle E!, x^P \rangle$ )
  using oth-class-taut-4-b [equiv-sym] by auto
qed

lemma prop-in-f-3-c:
  [ $\neg(\text{Indiscriminate } (O!))$  in  $v$ ]
proof (rule reductio-aa-2)
  show [ $\neg(\forall x . \langle O!, x^P \rangle)$  in  $v$ ]
  using a-objects-exist-2 [THEN qml-2 [axiom-instance, deduction]]
  by blast
next
  assume [ $\text{Indiscriminate } O!$  in  $v$ ]
  thus [ $(\forall x . \langle O!, x^P \rangle)$  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:
  [ $\neg(\text{Indiscriminate } (A!))$  in  $v$ ]
proof (rule reductio-aa-2)
  show [ $\neg(\forall x . \langle A!, x^P \rangle)$  in  $v$ ]
  using o-objects-exist-3 [THEN qml-2 [axiom-instance, deduction]]
  by blast
next
  assume [ $\text{Indiscriminate } A!$  in  $v$ ]
  thus [ $(\forall x . \langle A!, x^P \rangle)$  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*:
 $[\neg(\text{Propositional } E!) \text{ in } v]$
using *prop-in-thm[deduction]* *prop-in-f-3-a modus-tollens-1 CP*
by *meson*

lemma *prop-in-f-4-b*:
 $[\neg(\text{Propositional } (E!^-)) \text{ in } v]$
using *prop-in-thm[deduction]* *prop-in-f-3-b modus-tollens-1 CP*
by *meson*

lemma *prop-in-f-4-c*:
 $[\neg(\text{Propositional } (O!)) \text{ in } v]$
using *prop-in-thm[deduction]* *prop-in-f-3-c modus-tollens-1 CP*
by *meson*

lemma *prop-in-f-4-d*:
 $[\neg(\text{Propositional } (A!)) \text{ in } v]$
using *prop-in-thm[deduction]* *prop-in-f-3-d modus-tollens-1 CP*
by *meson*

lemma *prop-prop-nec-1*:
 $[\Diamond(\exists p . F = (\lambda x . p)) \rightarrow (\exists p . F = (\lambda x . p)) \text{ in } v]$
proof (*rule CP*)
 assume $[\Diamond(\exists p . F = (\lambda x . p)) \text{ in } v]$
 hence $[\exists p . \Diamond(F = (\lambda x . p)) \text{ in } v]$
 using *BF \Diamond [deduction]* **by** *auto*
 then obtain *p* **where** $[\Diamond(F = (\lambda x . p)) \text{ in } v]$
 by (*rule $\exists E$*)
 hence $[\Diamond(\Box(\forall x . \{x^P, F\} \equiv \{x^P, \lambda x . p\})) \text{ in } v]$
 unfolding *identity-defs* .
 hence $[\Box(\forall x . \{x^P, F\} \equiv \{x^P, \lambda x . p\}) \text{ in } v]$
 using *5 \Diamond [deduction]* **by** *auto*
 hence $[(F = (\lambda x . p)) \text{ in } v]$
 unfolding *identity-defs* .
 thus $[\exists p . (F = (\lambda x . p)) \text{ in } v]$
 by *PLM-solver*
qed

lemma *prop-prop-nec-2*:
 $[(\forall p . F \neq (\lambda x . p)) \rightarrow \Box(\forall p . F \neq (\lambda x . p)) \text{ in } v]$
apply (*PLM-subst-method*
 $\neg(\exists p . (F = (\lambda x . p)))$
 $(\forall p . \neg(F = (\lambda x . p)))$)
using *cqt-further-4* **apply** *blast*
apply (*PLM-subst-method*
 $\neg\Diamond(\exists p . F = (\lambda x . p))$
 $\Box\neg(\exists p . F = (\lambda x . p))$)
using *KBasic2-4[equiv-sym]* *prop-prop-nec-1*
 contraposition-1 **by** *auto*

lemma *prop-prop-nec-3*:
 $[(\exists p . F = (\lambda x . p)) \rightarrow \Box(\exists p . F = (\lambda x . p)) \text{ in } v]$
using *prop-prop-nec-1* *derived-S5-rules-1-b* **by** *simp*

lemma *prop-prop-nec-4*:
 $[\Diamond(\forall p . F \neq (\lambda x . p)) \rightarrow (\forall p . F \neq (\lambda x . p))] \text{ in } v]$
using *prop-prop-nec-2 derived-S5-rules-2-b* **by** *simp*

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

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

10 Possible Worlds

locale *PossibleWorlds* = *PLM*
begin

10.1 Definitions

definition *Situation* **where**
 $Situation\ x \equiv \langle A!, x \rangle \ \& \ (\forall F . \llbracket x, F \rrbracket \rightarrow Propositional\ F)$

definition *EncodeProposition* (*infixl* Σ 70) **where**
 $x\Sigma p \equiv \langle A!, x \rangle \ \& \ \llbracket x, \lambda x . p \rrbracket$

definition *TrueInSituation* (*infixl* \models 10) **where**
 $x \models p \equiv Situation\ x \ \& \ x\Sigma p$

definition *PossibleWorld* **where**
 $PossibleWorld\ x \equiv Situation\ x \ \& \ \Diamond(\forall p . x\Sigma p \equiv p)$

10.2 Auxiliary Lemmata

lemma *possit-sit-1*:
 $[Situation(x^P) \equiv \Box(Situation(x^P)) \text{ in } v]$
proof (rule $\equiv I$; rule *CP*)
 assume $[Situation(x^P) \text{ in } v]$
 hence 1: $[\Box(A!, x^P) \ \& \ (\forall F. \{x^P, F\} \rightarrow Propositional\ F) \text{ in } v]$
 unfolding *Situation-def* by *auto*
 have $[\Box(A!, x^P) \text{ in } v]$
 using 1[*conj1*, *THEN oa-facts-2[deduction]*] .
 moreover have $[\Box(\forall F. \{x^P, F\} \rightarrow Propositional\ F) \text{ in } v]$
 using 1[*conj2*] unfolding *Propositional-def*
 by (rule *enc-prop-nec-2[deduction]*)
 ultimately show $[\Box Situation(x^P) \text{ in } v]$
 unfolding *Situation-def*
 apply *cut-tac* apply (rule *KBasic-3[equiv-rl]*)
 by (rule *intro-elim-1*)
 next
 assume $[\Box Situation(x^P) \text{ in } v]$
 thus $[Situation(x^P) \text{ in } v]$
 using *qml-2[axiom-instance, deduction]* by *auto*
 qed

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

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

lemma *PossWorldAux*:
 $[(\Box(A!, x^P) \ \& \ (\forall F. (\{x^P, F\} \equiv (\exists p. p \ \& \ (F = (\lambda x. p))))))]$

```

→ (PossibleWorld (xP)) in v]
proof (rule CP)
  assume DefX: [(A!, xP) & (∀ F . (xP, F) ≡
    (∃ p . p & (F = (λ x . p)))) in v]

  have [Situation (xP) in v]
  proof -
    have [(A!, xP) in v]
    using DefX[conj1] .
    moreover have [(∀ F . (xP, F) → Propositional F) in v]
    proof (rule ∀I; rule CP)
      fix F
      assume [(xP, F) in v]
      moreover have [(xP, F) ≡ (∃ p . p & (F = (λ x . p))) in v]
      using DefX[conj2] cqt-1[axiom-instance, deduction] by auto
      ultimately have [(∃ p . p & (F = (λ x . p))) in v]
      using ≡E(1) by blast
      then obtain p where [p & (F = (λ x . p)) in v]
      by (rule ∃E)
      hence [(F = (λ x . p)) in v]
      by (rule &E(2))
      hence [(∃ p . (F = (λ x . p))) in v]
      by PLM-solver
      thus [Propositional F in v]
      unfolding Propositional-def .
    qed
    ultimately show [Situation (xP) in v]
    unfolding Situation-def by (rule &I)
  qed

  moreover have [◇(∀ p. xP Σ p ≡ p) in v]
  unfolding EncodeProposition-def
  proof (rule TBasic[deduction]; rule ∀I)
    fix q
    have EncodeLambda:
      [(xP, λx. q) ≡ (∃ p . p & ((λx. q) = (λ x . p))) in v]
      using DefX[conj2] by (rule cqt-1[axiom-instance, deduction])
    moreover {
      assume [q in v]
      moreover have [(λx. q) = (λ x . q) in v]
      using id-eq-prop-prop-1 by auto
      ultimately have [q & ((λx. q) = (λ x . q)) in v]
      by (rule &I)
      hence [∃ p . p & ((λx. q) = (λ x . p)) in v]
      by PLM-solver
      moreover have [(A!, xP) in v]
      using DefX[conj1] .
      ultimately have [(A!, xP) & (xP, λx. q) in v]
      using EncodeLambda[equiv-rl] &I by auto
    }
    moreover {
      assume [(A!, xP) & (xP, λx. q) in v]
      hence [(xP, λx. q) in v]
      using &E(2) by auto
      hence [∃ p . p & ((λx. q) = (λ x . p)) in v]

```

```

    using EncodeLambda[equiv-lr] by auto
  then obtain p where p-and-lambda-q-is-lambda-p:
    [p & ((λx. q) = (λ x . p)) in v]
    by (rule ∃ E)
  have [(λ x . p), xP] ≡ p in v]
    apply (rule beta-C-meta-1)
    by (rule IsPropositional-intros)+
  hence [(λ x . p), xP] in v]
    using p-and-lambda-q-is-lambda-p[conj1] ≡E(2) by auto
  hence [(λ x . q), xP] in v]
    using p-and-lambda-q-is-lambda-p[conj2, THEN id-eq-prop-prop-2[deduction]]
    l-identity[axiom-instance, deduction, deduction] by fast
  moreover have [(λ x . q), xP] ≡ q in v]
    apply (rule beta-C-meta-1) by (rule IsPropositional-intros)+
  ultimately have [q in v]
    using ≡E(1) by blast
}
ultimately show [(λ!,xP) & (λx. q)] ≡ q in v]
  using &I ≡I CP by auto
qed

ultimately show [PossibleWorld (xP) in v]
  unfolding PossibleWorld-def by (rule &I)
qed

```

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

theorem *SemanticPossibleWorldForSyntacticPossibleWorlds*:

$\forall x . [PossibleWorld (x^P) \text{ in } w] \longrightarrow$
 $(\exists v . \forall p . [p \text{ in } v] \longleftrightarrow [(x^P \models p) \text{ in } w])$

proof

fix x

{

assume *PossWorldX*: $[PossibleWorld (x^P) \text{ in } w]$

hence *SituationX*: $[Situation (x^P) \text{ in } w]$

unfolding *PossibleWorld-def* **apply** *cut-tac* **by** *PLM-solver*

have *PossWorldExpanded*:

$[(\lambda!,x^P) \& (\forall F. \llbracket x^P, F \rrbracket \rightarrow (\exists p. F = (\lambda x. p)))]$

$\& \Diamond(\forall p. (\lambda!,x^P) \& \llbracket x^P, \lambda x. p \rrbracket \equiv p) \text{ in } w]$

using *PossWorldX*

unfolding *PossibleWorld-def Situation-def*

Propositional-def EncodeProposition-def .

have *AbstractX*: $[(\lambda!,x^P) \text{ in } w]$

using *PossWorldExpanded*[*conj1*,*conj1*] .

have $[\Diamond(\forall p. \llbracket x^P, \lambda x. p \rrbracket \equiv p) \text{ in } w]$

apply (*PLM-subst1-method*

$\lambda p. (\lambda!,x^P) \& \llbracket x^P, \lambda x. p \rrbracket$

$\lambda p . \llbracket x^P, \lambda x. p \rrbracket$)

subgoal using *PossWorldExpanded*[*conj1*,*conj1*,*THEN oa-facts-2*[*deduction*]]

using *Semantics.T6* **apply** *cut-tac* **by** *PLM-solver*

using *PossWorldExpanded*[*conj2*] .

```

hence  $\exists v. \forall p. ([\![x^P, \lambda x. p]\!] \text{ in } v)$ 
       $= [p \text{ in } v]$ 
unfolding diamond-def equiv-def conj-def
apply (simp add: Semantics.T4 Semantics.T6 Semantics.T5
        Semantics.T8)
by auto

then obtain v where PropsTrueInSemWorld:
 $\forall p. ([\![x^P, \lambda x. p]\!] \text{ in } v) = [p \text{ in } v]$ 
by auto
{
  fix p
  {
    assume  $[(x^P) \models p] \text{ in } w$ 
    hence  $[(x^P) \models p] \text{ in } v$ 
    using TrueInWorldNec[equiv-lr] Semantics.T6 by simp
    hence  $[Situation(x^P) \ \& \ (\![A!, x^P]\!) \ \& \ [\![x^P, \lambda x. p]\!] \text{ in } v]$ 
    unfolding TrueInSituation-def EncodeProposition-def .
    hence  $[\![x^P, \lambda x. p]\!] \text{ in } v]$ 
    using &E(2) by blast
    hence  $[p \text{ in } v]$ 
    using PropsTrueInSemWorld by blast
  }
  moreover {
    assume  $[p \text{ in } v]$ 
    hence  $[\![x^P, \lambda x. p]\!] \text{ in } v]$ 
    using PropsTrueInSemWorld by blast
    hence  $[(x^P) \models p] \text{ in } v]$ 
apply cut-tac unfolding TrueInSituation-def EncodeProposition-def
apply (rule &I) using SituationX[THEN possit-sit-1[equiv-lr]]
    subgoal using Semantics.T6 by auto
    apply (rule &I)
    subgoal using AbstractX[THEN oa-facts-2[deduction]]
    using Semantics.T6 by auto
    by assumption
    hence  $[\Box((x^P) \models p) \text{ in } v]$ 
    using TrueInWorldNec[equiv-lr] by simp
    hence  $[(x^P) \models p \text{ in } w]$ 
    using Semantics.T6 by simp
  }
  ultimately have  $[p \text{ in } v] \longleftrightarrow [(x^P) \models p \text{ in } w]$ 
  by auto
}
hence  $(\exists v . \forall p . [p \text{ in } v] \longleftrightarrow [(x^P) \models p \text{ in } w])$ 
by blast
}
thus  $[PossibleWorld(x^P) \text{ in } w] \longrightarrow$ 
 $(\exists v. \forall p . [p \text{ in } v] \longleftrightarrow [(x^P) \models p \text{ in } w])$ 
by blast
qed

```

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

theorem *SyntacticPossibleWorldForSemanticPossibleWorlds*:

$\forall v . \exists x . [PossibleWorld (x^P) \text{ in } w] \wedge$
 $(\forall p . [p \text{ in } v] \longleftrightarrow [(x^P) \models p] \text{ in } w])$

proof

fix v

have $[\exists x . (A!, x^P) \& (\forall F . (\llbracket x^P, F \rrbracket \equiv$
 $(\exists p . p \& (F = (\lambda x . p)))) \text{ in } v]$

using *A-objects*[*axiom-instance*] **by** *fast*

then obtain x **where** *DefX*:

$[\llbracket A!, x^P \rrbracket \& (\forall F . (\llbracket x^P, F \rrbracket \equiv (\exists p . p \& (F = (\lambda x . p)))) \text{ in } v]$
by (*rule* $\exists E$)

hence *PossWorldX*: $[PossibleWorld (x^P) \text{ in } v]$

using *PossWorldAux*[*deduction*] **by** *blast*

hence $[PossibleWorld (x^P) \text{ in } w]$

using *possworld-nec*[*equiv-lr*] *Semantics.T6* **by** *auto*

moreover have $(\forall p . [p \text{ in } v] \longleftrightarrow [(x^P) \models p] \text{ in } w)$

proof

fix q

{

assume $[q \text{ in } v]$

moreover have $[(\lambda x . q) = (\lambda x . q) \text{ in } v]$

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

ultimately have $[q \& (\lambda x . q) = (\lambda x . q) \text{ in } v]$

using $\&I$ **by** *auto*

hence $[(\exists p . p \& ((\lambda x . q) = (\lambda x . p))) \text{ in } v]$

by *PLM-solver*

hence 4 : $[\llbracket x^P, (\lambda x . q) \rrbracket \text{ in } v]$

using *cqt-1*[*axiom-instance*, *deduction*, *OF DefX*[*conj2*], *equiv-lr*]
by *blast*

have $[(x^P) \models q] \text{ in } v]$

unfolding *TrueInSituation-def* **apply** (*rule* $\&I$)

using *PossWorldX* **unfolding** *PossibleWorld-def*

using $\&E(1)$ **apply** *blast*

unfolding *EncodeProposition-def* **apply** (*rule* $\&I$)

using *DefX*[*conj1*] **apply** *simp*

using 4 .

hence $[(x^P) \models q] \text{ in } w]$

using *TrueInWorldNecc*[*equiv-lr*] *Semantics.T6* **by** *auto*

}

moreover {

assume $[(x^P) \models q] \text{ in } w]$

hence $[(x^P) \models q] \text{ in } v]$

using *TrueInWorldNecc*[*equiv-lr*] *Semantics.T6*

by *auto*

hence $[\llbracket x^P, (\lambda x . q) \rrbracket \text{ in } v]$

unfolding *TrueInSituation-def* *EncodeProposition-def*

using $\&E(2)$ **by** *blast*

hence $[(\exists p . p \& ((\lambda x . q) = (\lambda x . p))) \text{ in } v]$

using *cqt-1*[*axiom-instance*, *deduction*, *OF DefX*[*conj2*], *equiv-lr*]
by *blast*

then obtain p **where** 4 :

```

      [(p & ((λ x . q) = (λ x . p))) in v]
    by (rule ∃ E)
  have [(λ (λ x . p), xP) ≡ p in v] apply (rule beta-C-meta-1)
    by (rule IsPropositional-intros)+
  hence [(λ (λ x . q), xP) ≡ p in v]
    using l-identity[where β=(λ x . q) and α=(λ x . p),
      axiom-instance, deduction, deduction]
    using 4[conj2, THEN id-eq-prop-prop-2[deduction]] by meson
  hence [(λ (λ x . q), xP) in v] using 4[conj1] ≡E(2) by blast
  moreover have [(λ (λ x . q), xP) ≡ q in v]
    apply (rule beta-C-meta-1)
    by (rule IsPropositional-intros)+
  ultimately have [q in v]
    using ≡E(1) by blast
}
ultimately show [q in v] ⟷ [(xP) ⊨ q in w]
  by blast
qed
ultimately show ∃ x . [PossibleWorld (xP) in w]
  ∧ (∀ p . [p in v] ⟷ [(xP) ⊨ p in w])
  by auto
qed
end

```

11 Sanity Tests

11.1 Consistency

```

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

```

11.2 Intensionality

```

context
begin
  interpretation MetaSolver.

  lemma [(λy. (q ∨ ¬q)) = (λy. (p ∨ ¬p)) in v]
    unfolding identity-Π1-def
    apply (rule Eq1I) 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 ⇒ bool) × i = 4,
      card (i ⇒ bool) × (i ⇒ bool) × i = 4,
      card v = 2, verbose, show-all, debug]
    oops — Countermodel by Nitpick
  lemma [(λy. (p ∨ q)) = (λy. (q ∨ p)) in v]
    unfolding identity-Π1-def

```

```

apply (rule  $Eq_1I$ ) apply (simp add: meta-defs)
nitpick[expect = genuine, user-axioms=true,
        sat-solver = MiniSat-JNI,
        card i = 2, card j = 2, card  $\sigma$  = 1,
        card  $\omega$  = 1, card ( $i \Rightarrow \text{bool}$ )  $\times$   $i$  = 4,
        card ( $i \Rightarrow \text{bool}$ )  $\times$  ( $i \Rightarrow \text{bool}$ )  $\times$   $i$  = 4,
        card v = 2, verbose, show-all, debug]
oops — Countermodel by Nitpick
end

```

11.3 Concreteness coindices with Object Domains

```

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

```

11.4 Justification for Meta-Logical Axioms

```

context
begin

```

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

```

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

```

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

```

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

```

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

```

  private lemma PossiblyContingentObjectExistsCheck:
     $[\neg(\Box(\forall x. \langle E!, x^P \rangle \rightarrow \Box \langle E!, x^P \rangle)) \text{ in } v]$ 

```



```

    apply (simp add: meta-defs forall-ν-def meta-aux split: ν.split v.split)
    using PossiblyContingentObjectExistsAxiom
    by (metis ν.simps(5) νν-def v.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 v.split)
  by (metis v.exhaust v.simps(5) v.simps(6))

```

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

```

private lemma PossiblyNoContingentObjectExistsCheck:
  [¬(□(¬(∀ x. (⊨E!,xP) → □(⊨E!,xP)))) in v]
  apply (simp add: meta-defs forall-ν-def meta-aux split: ν.split v.split)
  using PossiblyNoContingentObjectExistsAxiom by blast
private lemma PossiblyNoContingentObjectExists
  using PossiblyNoContingentObjectExistsCheck
  apply (auto simp: meta-defs forall-ν-def meta-aux split: ν.split v.split)
  by (metis v.simps(5) νν-νν-id)
end

```

11.5 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:
  [∀ x . □(⊨F,xP) ≡ ⊨G,xP) in v] ⟷
  ((λ x . (evalΠ1 F) x dj) = (λ x . (evalΠ1 G) x dj))
proof
  interpret MetaSolver .
  interpret Semantics .
  assume 1: [∀ x . □(⊨F,xP) ≡ ⊨G,xP) in v]
  {
    fix v
    fix y
    obtain x where y-def: y = νν x by (metis νν-νν-id)
    have (∃ r o1. Some r = d1 F ∧ Some o1 = dκ (xP) ∧ o1 ∈ ex1 r
v) =
      (∃ r o1. Some r = d1 G ∧ Some o1 = dκ (xP) ∧ o1 ∈ ex1 r v)
      using 1 apply cut-tac by meta-solver
    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
    moreover have Some x = dκ (xP)
    using dκ-proper by simp
    ultimately have (x ∈ ex1 r v) = (x ∈ ex1 s v)
    by (metis option.inject)
    hence (evalΠ1 F) y dj v = (evalΠ1 G) y dj v
  }

```

```

    using r-def s-def y-def by (simp add: d1.rep-eq ex1-def)
  }
  thus ( $\lambda x. \text{eval}\Pi_1 F x dj$ ) = ( $\lambda x. \text{eval}\Pi_1 G x dj$ )
    by auto
next
interpret MetaSolver .
interpret Semantics .
assume 1: ( $\lambda x. \text{eval}\Pi_1 F x dj$ ) = ( $\lambda x. \text{eval}\Pi_1 G x dj$ )
{
  fix y v
  obtain x where x-def:  $x = \nu v y$ 
    by simp
  hence  $\text{eval}\Pi_1 F x dj = \text{eval}\Pi_1 G x dj$ 
    using 1 by metis
  moreover obtain r where r-def:  $\text{Some } r = d_1 F$ 
    unfolding d1-def by auto
  moreover obtain s where s-def:  $\text{Some } s = d_1 G$ 
    unfolding d1-def by auto
  ultimately have  $(y \in \text{ex1 } r v) = (y \in \text{ex1 } s v)$ 
    by (simp add: d1.rep-eq ex1-def  $\nu v$ - $\nu v$ -id x-def)
  hence  $(\llbracket F, y^P \rrbracket \equiv \llbracket G, y^P \rrbracket \text{ in } v)$ 
    apply cut-tac apply meta-solver
    using r-def s-def by (metis Semantics.d $\kappa$ -proper option.inject)
}
thus  $[\forall x. \Box(\llbracket F, x^P \rrbracket \equiv \llbracket G, x^P \rrbracket) \text{ in } v]$ 
  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  $[\forall x. \Box(\llbracket F, x^P \rrbracket \equiv \llbracket G, x^P \rrbracket) \text{ in } v]$ 
  shows  $[F = G \text{ in } v] \longleftrightarrow$ 
    ( $\forall s. s \neq dj \longrightarrow (\forall x. (\text{eval}\Pi_1 F) x s = (\text{eval}\Pi_1 G) x s)$ )
proof
  interpret MetaSolver .
  assume  $[F = G \text{ in } v]$ 
  hence  $F = G$ 
    apply cut-tac unfolding identity- $\Pi_1$ -def by meta-solver
  thus  $\forall s. s \neq dj \longrightarrow (\forall x. \text{eval}\Pi_1 F x s = \text{eval}\Pi_1 G x s)$ 
    by auto
next
interpret MetaSolver .
assume  $\forall s. s \neq dj \longrightarrow (\forall x. \text{eval}\Pi_1 F x s = \text{eval}\Pi_1 G x s)$ 
moreover have  $((\lambda x. (\text{eval}\Pi_1 F) x dj) = (\lambda x. (\text{eval}\Pi_1 G) x dj))$ 
  using assms mat-eq-is-eq-dj by auto
ultimately have  $\forall s x. \text{eval}\Pi_1 F x s = \text{eval}\Pi_1 G x s$ 
  by metis
hence  $\text{eval}\Pi_1 F = \text{eval}\Pi_1 G$ 
  by blast
hence  $F = G$ 
  by (metis eval $\Pi_1$ -inverse)
thus  $[F = G \text{ in } v]$ 

```

unfolding *identity- Π_1 -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 $\forall F x s . (eval\Pi_1 F) x s = (eval\Pi_1 F) x dj$
shows $[\forall x . \Box(\llbracket F, x^P \rrbracket \equiv \llbracket G, x^P \rrbracket) \text{ in } v] \longleftrightarrow [F = G \text{ in } v]$
by (*metis (no-types) MetaSolver.Eq₁S assms identity- Π_1 -def*
mat-eq-is-eq-dj mat-eq-is-eq-if-eq-forall-j)

end