

Embedding of the Theory of Abstract Objects in Isabelle/HOL

Daniel Kirchner

May 30, 2017

Abstract

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

Contents

1	Representation Layer	3
1.1	Primitives	3
1.2	Derived Types	3
1.3	Individual Terms and Definite Descriptions	4
1.4	Mapping from Individuals to Urelements	4
1.5	Exemplification of n-place-Relations.	4
1.6	Encoding	4
1.7	Connectives and Quantifiers	4
1.8	Lambda Expressions	5
1.9	Proper Maps	5
1.10	Validity	5
1.11	Concreteness	5
1.12	Collection of Meta-Definitions	6
1.13	Auxiliary Lemmata	6
2	Semantic Abstraction	7
2.1	Semantics	7
2.1.1	Semantic Domains	7
2.1.2	Denotation Functions	7
2.1.3	Actual World	7
2.1.4	Exemplification Extensions	7
2.1.5	Encoding Extensions	8
2.1.6	Collection of Semantic Definitions	8
2.1.7	Truth Conditions of Exemplification Formulas	8
2.1.8	Truth Conditions of Encoding Formulas	8
2.1.9	Truth Conditions of Complex Formulas	8
2.1.10	Denotations of Descriptions	9
2.1.11	Denotations of Lambda Expressions	9
2.1.12	Auxiliary Lemmas	10
2.2	Introduction Rules for Proper Maps	10
2.3	Validity Syntax	12
3	General Quantification	12
3.1	Type Class	12
3.2	Instantiations	13
4	Basic Definitions	13
4.1	Derived Connectives	13
4.2	Abstract and Ordinary Objects	14
4.3	Identity Definitions	14

5	MetaSolver	14
5.1	Rules for Implication	15
5.2	Rules for Negation	15
5.3	Rules for Conjunction	15
5.4	Rules for Equivalence	15
5.5	Rules for Disjunction	15
5.6	Rules for Necessity	15
5.7	Rules for Possibility	16
5.8	Rules for Quantification	16
5.8.1	Rules for Existence	16
5.9	Rules for Actuality	16
5.10	Rules for Encoding	16
5.11	Rules for Exemplification	16
5.11.1	Zero-place Relations	16
5.11.2	One-Place Relations	17
5.11.3	Two-Place Relations	17
5.11.4	Three-Place Relations	17
5.12	Rules for Being Ordinary	17
5.13	Rules for Being Abstract	18
5.14	Rules for Definite Descriptions	19
5.15	Rules for Identity	19
5.15.1	Ordinary Objects	19
5.15.2	Individuals	20
5.15.3	One-Place Relations	21
5.15.4	Two-Place Relations	22
5.15.5	Three-Place Relations	22
5.15.6	Propositions	23
6	General Identity	24
6.1	Type Classes	24
6.2	Instantiations	24
6.3	New Identity Definitions	25
7	The Axioms of PLM	26
7.1	Closures	26
7.2	Axioms for Negations and Conditionals	27
7.3	Axioms of Identity	27
7.4	Axioms of Quantification	27
7.5	Axioms of Actuality	28
7.6	Axioms of Necessity	28
7.7	Axioms of Necessity and Actuality	29
7.8	Axioms of Descriptions	29
7.9	Axioms for Complex Relation Terms	30
7.10	Axioms of Encoding	31
8	Definitions	32
8.1	Property Negations	32
8.2	Noncontingent and Contingent Relations	32
8.3	Null and Universal Objects	33
8.4	Propositional Properties	33
8.5	Indiscriminate Properties	33
8.6	Miscellaneous	33
9	The Deductive System PLM	33
9.1	Automatic Solver	33
9.2	Modus Ponens	34
9.3	Axioms	34
9.4	(Modally Strict) Proofs and Derivations	34
9.5	GEN and RN	34
9.6	Negations and Conditionals	34
9.7	Identity	41
9.8	Quantification	46
9.9	Actuality and Descriptions	48
9.10	Necessity	59

9.11 The Theory of Relations	75
9.12 The Theory of Objects	99
9.13 Propositional Properties	115
10 Possible Worlds	119
10.1 Definitions	120
10.2 Auxiliary Lemmas	120
10.3 For every syntactic Possible World there is a semantic Possible World	122
10.4 For every semantic Possible World there is a syntactic Possible World	123
11 Artificial Theorems	124
12 Sanity Tests	126
12.1 Consistency	126
12.2 Intensionality	127
12.3 Concreteness coindices with Object Domains	127
12.4 Justification for Meta-Logical Axioms	127
12.5 Relations in the Meta-Logic	128
12.6 Lambda Expressions	129
13 Paradox	130
13.1 Auxiliary Lemmas	130
13.2 Fake β -Conversion using Description Backdoor	131
13.3 Resulting Paradox	132
13.4 Original Version of the Paradox	132

1 Representation Layer

1.1 Primitives

typedecl i — possible worlds

typedecl j — states

consts $dw :: i$ — actual world

consts $dj :: j$ — actual state

typedecl ω — ordinary objects

typedecl σ — special urelements

datatype $v = \omega v \ \omega \mid \sigma v \ \sigma$ — urelements

1.2 Derived Types

typedef $o = UNIV :: (j \Rightarrow i \Rightarrow bool)$ *set*

morphisms $eval_o \ make_o \ ..$ — truth values

type-synonym $\Pi_0 = o$ — zero place relations

typedef $\Pi_1 = UNIV :: (v \Rightarrow j \Rightarrow i \Rightarrow bool)$ *set*

morphisms $eval_{\Pi_1} \ make_{\Pi_1} \ ..$ — one place relations

typedef $\Pi_2 = UNIV :: (v \Rightarrow v \Rightarrow j \Rightarrow i \Rightarrow bool)$ *set*

morphisms $eval_{\Pi_2} \ make_{\Pi_2} \ ..$ — two place relations

typedef $\Pi_3 = UNIV :: (v \Rightarrow v \Rightarrow v \Rightarrow j \Rightarrow i \Rightarrow bool)$ *set*

morphisms $eval_{\Pi_3} \ make_{\Pi_3} \ ..$ — three place relations

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

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

typedef $\kappa = UNIV :: (\nu \ option)$ *set*

morphisms $eval_{\kappa} \ make_{\kappa} \ ..$ — individual terms

setup-lifting *type-definition-o*

setup-lifting *type-definition- κ*

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

1.3 Individual Terms and Definite Descriptions

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

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

1.4 Mapping from Individuals to Urelements

consts $\alpha\sigma :: \alpha \Rightarrow \sigma$
axiomatization **where** $\alpha\sigma\text{-surj} :: \textit{surj} \text{ } \alpha\sigma$
definition $\nu\nu :: \nu \Rightarrow \nu \text{ where } \nu\nu \equiv \textit{case-}\nu \text{ } \omega\nu \text{ } (\sigma\nu \circ \alpha\sigma)$

1.5 Exemplification of n-place-Relations.

lift-definition $\textit{exe0} :: \Pi_0 \Rightarrow o \text{ (}\llbracket - \rrbracket \text{) is } \textit{id} \text{ } .$
lift-definition $\textit{exe1} :: \Pi_1 \Rightarrow \kappa \Rightarrow o \text{ (}\llbracket -, - \rrbracket \text{) is}$
 $\lambda F x s w . (\textit{proper} x) \wedge F (\nu\nu (\textit{rep} x)) s w \text{ } .$
lift-definition $\textit{exe2} :: \Pi_2 \Rightarrow \kappa \Rightarrow \kappa \Rightarrow o \text{ (}\llbracket -, -, - \rrbracket \text{) is}$
 $\lambda F x y s w . (\textit{proper} x) \wedge (\textit{proper} y) \wedge$
 $F (\nu\nu (\textit{rep} x)) (\nu\nu (\textit{rep} y)) s w \text{ } .$
lift-definition $\textit{exe3} :: \Pi_3 \Rightarrow \kappa \Rightarrow \kappa \Rightarrow \kappa \Rightarrow o \text{ (}\llbracket -, -, -, - \rrbracket \text{) is}$
 $\lambda F x y z s w . (\textit{proper} x) \wedge (\textit{proper} y) \wedge (\textit{proper} z) \wedge$
 $F (\nu\nu (\textit{rep} x)) (\nu\nu (\textit{rep} y)) (\nu\nu (\textit{rep} z)) s w \text{ } .$

1.6 Encoding

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

1.7 Connectives and Quantifiers

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

lift-definition $\textit{not} :: o \Rightarrow o \text{ (}\neg \text{ } [54] \text{ } 70) \text{ is}$
 $\lambda p s w . s = dj \wedge \neg p \text{ } dj \text{ } w \vee s \neq dj \wedge (I\text{-NOT} s (p s) w) \text{ } .$
lift-definition $\textit{impl} :: o \Rightarrow o \Rightarrow o \text{ (infixl } \rightarrow \text{ } 51) \text{ is}$
 $\lambda p q s w . s = dj \wedge (p \text{ } dj \text{ } w \longrightarrow q \text{ } dj \text{ } w) \vee s \neq dj \wedge (I\text{-IMPL} s (p s) (q s) w) \text{ } .$
lift-definition $\textit{forall}_\nu :: (\nu \Rightarrow o) \Rightarrow o \text{ (binder } \forall_\nu \text{ } [8] \text{ } 9) \text{ is}$
 $\lambda \varphi s w . \forall x :: \nu . (\varphi x) s w \text{ } .$
lift-definition $\textit{forall}_0 :: (\Pi_0 \Rightarrow o) \Rightarrow o \text{ (binder } \forall_0 \text{ } [8] \text{ } 9) \text{ is}$
 $\lambda \varphi s w . \forall x :: \Pi_0 . (\varphi x) s w \text{ } .$
lift-definition $\textit{forall}_1 :: (\Pi_1 \Rightarrow o) \Rightarrow o \text{ (binder } \forall_1 \text{ } [8] \text{ } 9) \text{ is}$
 $\lambda \varphi s w . \forall x :: \Pi_1 . (\varphi x) s w \text{ } .$
lift-definition $\textit{forall}_2 :: (\Pi_2 \Rightarrow o) \Rightarrow o \text{ (binder } \forall_2 \text{ } [8] \text{ } 9) \text{ is}$
 $\lambda \varphi s w . \forall x :: \Pi_2 . (\varphi x) s w \text{ } .$
lift-definition $\textit{forall}_3 :: (\Pi_3 \Rightarrow o) \Rightarrow o \text{ (binder } \forall_3 \text{ } [8] \text{ } 9) \text{ is}$
 $\lambda \varphi s w . \forall x :: \Pi_3 . (\varphi x) s w \text{ } .$
lift-definition $\textit{forall}_o :: (o \Rightarrow o) \Rightarrow o \text{ (binder } \forall_o \text{ } [8] \text{ } 9) \text{ is}$
 $\lambda \varphi s w . \forall x :: o . (\varphi x) s w \text{ } .$
lift-definition $\textit{box} :: o \Rightarrow o \text{ (}\Box \text{ } [62] \text{ } 63) \text{ is}$
 $\lambda p s w . \forall v . p s v \text{ } .$

lift-definition $actual :: o \Rightarrow o$ (\mathcal{A} - [64] 65) **is**
 $\lambda p s w . p s dw .$

Remark 1. *The connectives behave classically if evaluated for the actual state dj , whereas their behavior is governed by uninterpreted constants for any other state.*

1.8 Lambda Expressions

Remark 2. *Lambda expressions have to convert maps from individuals to propositions to relations that are represented by maps from urelements to truth values.*

lift-definition $lambdabinder0 :: o \Rightarrow \Pi_0 (\lambda^0)$ **is** $id .$

lift-definition $lambdabinder1 :: (\nu \Rightarrow o) \Rightarrow \Pi_1$ ($binder \lambda [8] 9$) **is**
 $\lambda \varphi u s w . \exists x . \nu v x = u \wedge \varphi x s w .$

lift-definition $lambdabinder2 :: (\nu \Rightarrow \nu \Rightarrow o) \Rightarrow \Pi_2 (\lambda^2)$ **is**

$\lambda \varphi u v s w . \exists x y . \nu v x = u \wedge \nu v y = v \wedge \varphi x y s w .$

lift-definition $lambdabinder3 :: (\nu \Rightarrow \nu \Rightarrow \nu \Rightarrow o) \Rightarrow \Pi_3 (\lambda^3)$ **is**

$\lambda \varphi u v r s w . \exists x y z . \nu v x = u \wedge \nu v y = v \wedge \nu v z = r \wedge \varphi x y z s w .$

1.9 Proper Maps

Remark 3. *The embedding introduces the notion of proper maps from individual terms to propositions.*

Such a map is proper if and only if for all proper individual terms its truth evaluation in the actual state only depends on the urelements corresponding to the individuals the terms denote.

Proper maps are exactly those maps that - when used in as matrix of a lambda-expression - unconditionally allow beta-reduction.

lift-definition $IsProperInX :: (\kappa \Rightarrow o) \Rightarrow bool$ **is**

$\lambda \varphi . \forall x v . (\exists a . \nu v a = \nu v x \wedge (\varphi (a^P) dj v)) = (\varphi (x^P) dj v) .$

lift-definition $IsProperInXY :: (\kappa \Rightarrow \kappa \Rightarrow o) \Rightarrow bool$ **is**

$\lambda \varphi . \forall x y v . (\exists a b . \nu v a = \nu v x \wedge \nu v b = \nu v y$
 $\wedge (\varphi (a^P) (b^P) dj v)) = (\varphi (x^P) (y^P) dj v) .$

lift-definition $IsProperInXYZ :: (\kappa \Rightarrow \kappa \Rightarrow \kappa \Rightarrow o) \Rightarrow bool$ **is**

$\lambda \varphi . \forall x y z v . (\exists a b c . \nu v a = \nu v x \wedge \nu v b = \nu v y \wedge \nu v c = \nu v z$
 $\wedge (\varphi (a^P) (b^P) (c^P) dj v)) = (\varphi (x^P) (y^P) (z^P) dj v) .$

1.10 Validity

lift-definition $valid-in :: i \Rightarrow o \Rightarrow bool$ ($infixl \models 5$) **is**

$\lambda v \varphi . \varphi dj v .$

Remark 4. *A formula is considered semantically valid for a possible world, if it evaluates to True for the actual state dj and the given possible world.*

1.11 Concreteness

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

abbreviation (input) $OrdinaryObjectsPossiblyConcrete$ **where**

$OrdinaryObjectsPossiblyConcrete \equiv \forall x . \exists v . ConcreteInWorld x v$

abbreviation (input) $PossiblyContingentObjectExists$ **where**

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

abbreviation (input) $PossiblyNoContingentObjectExists$ **where**

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

axiomatization **where**

OrdinaryObjectsPossiblyConcreteAxiom:
OrdinaryObjectsPossiblyConcrete
and *PossiblyContingentObjectExistsAxiom:*
PossiblyContingentObjectExists
and *PossiblyNoContingentObjectExistsAxiom:*
PossiblyNoContingentObjectExists

Remark 5. Care has to be taken that the defined notion of concreteness coincides with the meta-logical distinction between abstract objects and ordinary objects. Furthermore the axioms about concreteness have to be satisfied. This is achieved by introducing an uninterpreted constant *ConcreteInWorld* that determines whether an ordinary object is concrete in a given possible world. This constant is axiomatized, such that all ordinary objects are possibly concrete, contingent objects possibly exist and possibly no contingent objects exist.

lift-definition *Concrete:: Π_1 (E!)* is
 $\lambda u s w . \text{case } u \text{ of } \omega v x \Rightarrow \text{ConcreteInWorld } x w \mid - \Rightarrow \text{False} .$

Remark 6. Concreteness of ordinary objects is now defined using this axiomatized uninterpreted constant. Abstract objects on the other hand are never concrete.

1.12 Collection of Meta-Definitions

named-theorems *meta-defs*

declare *not-def*[*meta-defs*] *impl-def*[*meta-defs*] *forall_v-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]

1.13 Auxiliary Lemmata

named-theorems *meta-aux*

declare *make κ -inverse*[*meta-aux*] *eval κ -inverse*[*meta-aux*]
make ω -inverse[*meta-aux*] *eval ω -inverse*[*meta-aux*]
make Π_1 -inverse[*meta-aux*] *eval Π_1 -inverse*[*meta-aux*]
make Π_2 -inverse[*meta-aux*] *eval Π_2 -inverse*[*meta-aux*]
make Π_3 -inverse[*meta-aux*] *eval Π_3 -inverse*[*meta-aux*]
lemma *$\nu\nu$ - $\omega\nu$ -is- $\omega\nu$* [*meta-aux*]: $\nu\nu (\omega\nu x) = \omega\nu x$ **by** (*simp add: $\nu\nu$ -def*)
lemma *rep-proper-id*[*meta-aux*]: $\text{rep } (x^P) = x$
by (*simp add: meta-aux $\nu\kappa$ -def rep-def*)
lemma *$\nu\kappa$ -proper*[*meta-aux*]: $\text{proper } (x^P)$
by (*simp add: meta-aux $\nu\kappa$ -def proper-def*)
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 *$\alpha\sigma$ -surj* **unfolding** *$\nu\nu$ -def surj-def*
by (*metis ν .simps(5) ν .simps(6) ν .exhaust comp-apply*)
lemma *lambda Π_1 -aux*[*meta-aux*]:
make Π_1 ($\lambda u s w . \exists x . \nu\nu x = u \wedge \text{eval}\Pi_1 F (\nu\nu x) s w) = F$
proof –
have $\bigwedge u s w \varphi . (\exists x . \nu\nu x = u \wedge \varphi (\nu\nu x) (s::j) (w::i)) \longleftrightarrow \varphi u s w$

```

    using  $\nu\nu$ -surj unfolding surj-def by metis
  thus ?thesis apply transfer by simp
qed
lemma lambda $\Pi_2$ -aux[meta-aux]:
  make $\Pi_2$  ( $\lambda u v s w. \exists x. \nu\nu x = u \wedge (\exists y. \nu\nu y = v \wedge \text{eval}\Pi_2 F (\nu\nu x) (\nu\nu y) s w)$ ) = F
proof -
  have  $\bigwedge u v (s :: j) (w :: i) \varphi .$ 
    ( $\exists x. \nu\nu x = u \wedge (\exists y. \nu\nu y = v \wedge \varphi (\nu\nu x) (\nu\nu y) s w)$ )
     $\longleftrightarrow \varphi u v s w$ 
  using  $\nu\nu$ -surj unfolding surj-def by metis
  thus ?thesis apply transfer by simp
qed
lemma lambda $\Pi_3$ -aux[meta-aux]:
  make $\Pi_3$  ( $\lambda u v r s w. \exists x. \nu\nu x = u \wedge (\exists y. \nu\nu y = v \wedge$ 
    ( $\exists z. \nu\nu z = r \wedge \text{eval}\Pi_3 F (\nu\nu x) (\nu\nu y) (\nu\nu z) s w))$ ) = F
proof -
  have  $\bigwedge u v r (s :: j) (w :: i) \varphi . \exists x. \nu\nu x = u \wedge (\exists y. \nu\nu y = v$ 
     $\wedge (\exists z. \nu\nu z = r \wedge \varphi (\nu\nu x) (\nu\nu y) (\nu\nu z) s w)) = \varphi u v r s w$ 
  using  $\nu\nu$ -surj unfolding surj-def by metis
  thus ?thesis apply transfer apply (rule ext)+ by metis
qed

```

2 Semantic Abstraction

2.1 Semantics

```

locale Semantics
begin
  named-theorems semantics

```

2.1.1 Semantic Domains

```

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  $W = i$ 

```

2.1.2 Denotation Functions

```

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

```

2.1.3 Actual World

```

definition  $w_0$  where  $w_0 \equiv dw$ 

```

2.1.4 Exemplification Extensions

```

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

```

2.1.5 Encoding Extensions

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

2.1.6 Collection of Semantic Definitions

named-theorems *semantics-defs*
declare $d_0\text{-def}[semantics-defs]$ $d_1\text{-def}[semantics-defs]$
 $d_2\text{-def}[semantics-defs]$ $d_3\text{-def}[semantics-defs]$
 $ex0\text{-def}[semantics-defs]$ $ex1\text{-def}[semantics-defs]$
 $ex2\text{-def}[semantics-defs]$ $ex3\text{-def}[semantics-defs]$
 $en\text{-def}[semantics-defs]$ $d_\kappa\text{-def}[semantics-defs]$
 $w_0\text{-def}[semantics-defs]$

2.1.7 Truth Conditions of Exemplification Formulas

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

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

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

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

2.1.8 Truth Conditions of Encoding Formulas

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

2.1.9 Truth Conditions of Complex Formulas

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

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

lemma $T6[semantics]$: $(w \models \Box \psi) = (\forall v . (v \models \psi))$

by (simp add: meta-defs meta-aux)

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

2.1.10 Denotations of Descriptions

lemma D3[semantics]:

$$d_\kappa (\iota x. \psi x) = (\text{if } (\exists x. (w_0 \models \psi x) \wedge (\forall y. (w_0 \models \psi y) \longrightarrow y = x))$$

$$\text{then } (\text{Some } (THE x. (w_0 \models \psi x))) \text{ else None})$$

 unfolding semantics-defs
 by (auto simp: meta-defs meta-aux)

2.1.11 Denotations of Lambda Expressions

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

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

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

lemma D5-1[semantics]:
 assumes IsProperInX 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 IsProperInX-def semantics-defs
 by (auto simp: meta-defs meta-aux rep-def proper-def $\nu\kappa.\text{abs-eq}$)

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

lemma D5-3[semantics]:
 assumes IsProperInXYZ φ
 shows $\bigwedge w o_1 o_2 o_3 r. \text{Some } r = d_3 (\lambda^3 (\lambda x y z. \varphi (x^P) (y^P) (z^P)))$
 $\wedge \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y \wedge \text{Some } o_3 = d_\kappa z$
 $\longrightarrow ((o_1, o_2, o_3) \in \text{ex3 } r w) = (w \models \varphi x y z)$
 using assms unfolding IsProperInXYZ-def semantics-defs

by (auto simp: meta-defs meta-aux rep-def proper-def $\nu\kappa$.abs-eq)

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

2.1.12 Auxiliary Lemmas

lemma propex_0 : $\exists r . \text{Some } r = d_0 F$
 unfolding d_0 -def by simp
lemma propex_1 : $\exists r . \text{Some } r = d_1 F$
 unfolding d_1 -def by simp
lemma propex_2 : $\exists r . \text{Some } r = d_2 F$
 unfolding d_2 -def by simp
lemma propex_3 : $\exists r . \text{Some } r = d_3 F$
 unfolding d_3 -def by simp
lemma d_κ -proper: $d_\kappa (u^P) = \text{Some } u$
 unfolding d_κ -def by (simp add: $\nu\kappa$ -def meta-aux)
lemma $\text{ConcretenessSemantics1}$:
 $\text{Some } r = d_1 E! \implies (\exists w . \omega\nu x \in \text{ex1 } r w)$
 unfolding semantics-defs apply transfer
 by (simp add: OrdinaryObjectsPossiblyConcreteAxiom $\nu\nu$ - $\omega\nu$ -is- $\omega\nu$)
lemma $\text{ConcretenessSemantics2}$:
 $\text{Some } r = d_1 E! \implies (x \in \text{ex1 } r w \longrightarrow (\exists y . x = \omega\nu y))$
 unfolding semantics-defs apply transfer apply simp
 by (metis ν .exhaust ν .exhaust ν .simps(6) no- $\alpha\omega$)
lemma d_0 -inject: $\bigwedge x y . d_0 x = d_0 y \implies x = y$
 unfolding d_0 -def by (simp add: eval0-inject)
lemma d_1 -inject: $\bigwedge x y . d_1 x = d_1 y \implies x = y$
 unfolding d_1 -def by (simp add: eval Π_1 -inject)
lemma d_2 -inject: $\bigwedge x y . d_2 x = d_2 y \implies x = y$
 unfolding d_2 -def by (simp add: eval Π_2 -inject)
lemma d_3 -inject: $\bigwedge x y . d_3 x = d_3 y \implies x = y$
 unfolding d_3 -def by (simp add: eval Π_3 -inject)
lemma d_κ -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 $\text{Some } o_1 = d_\kappa x \wedge \text{Some } o_1 = d_\kappa y$
 thus $x = y$ apply transfer by auto
qed
end

2.2 Introduction Rules for Proper Maps

Remark 7. Every map whose arguments only occur in exemplification expressions is proper.

named-theorems IsProper-intros

lemma $\text{IsProperInX-intro}[\text{IsProper-intros}]$:
 $\text{IsProperInX } (\lambda x . \chi$
 (* 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 *) $(\lambda F a b . \langle F, x, a, b \rangle) (\lambda F a b . \langle F, a, x, b \rangle)$
 $(\lambda F a b . \langle F, a, b, x \rangle))$
 unfolding IsProperInX-def
 by (auto simp: meta-defs meta-aux)

lemma $\text{IsProperInXY-intro}[\text{IsProper-intros}]$:
 $\text{IsProperInXY } (\lambda x y . \chi$
 (* only x *)
 (* one place *) $(\lambda F . \langle F, x \rangle)$


```

      (λ F . (⟦F,y,y,x⟧))
(* three place (x,x,x) *) (λ F . (⟦F,x,x,x⟧))
(* three place (y,y,y) *) (λ F . (⟦F,y,y,y⟧))
(* x and z *)
(* two place *) (λ F . (⟦F,x,z⟧)) (λ F . (⟦F,z,x⟧))
(* three place (x,z) *) (λ F a . (⟦F,x,z,a⟧)) (λ F a . (⟦F,x,a,z⟧))
      (λ F a . (⟦F,a,x,z⟧))
(* three place (z,x) *) (λ F a . (⟦F,z,x,a⟧)) (λ F a . (⟦F,z,a,x⟧))
      (λ F a . (⟦F,a,z,x⟧))
(* three place (x,x,z) *) (λ F . (⟦F,x,x,z⟧)) (λ F . (⟦F,x,z,x⟧))
      (λ F . (⟦F,z,x,x⟧))
(* three place (x,z,z) *) (λ F . (⟦F,x,z,z⟧)) (λ F . (⟦F,z,x,z⟧))
      (λ F . (⟦F,z,z,x⟧))
(* three place (x,x,x) *) (λ F . (⟦F,x,x,x⟧))
(* three place (z,z,z) *) (λ F . (⟦F,z,z,z⟧))
(* y and z *)
(* two place *) (λ F . (⟦F,y,z⟧)) (λ F . (⟦F,z,y⟧))
(* three place (y,z) *) (λ F a . (⟦F,y,z,a⟧)) (λ F a . (⟦F,y,a,z⟧))
      (λ F a . (⟦F,a,y,z⟧))
(* three place (z,y) *) (λ F a . (⟦F,z,y,a⟧)) (λ F a . (⟦F,z,a,y⟧))
      (λ F a . (⟦F,a,z,y⟧))
(* three place (y,y,z) *) (λ F . (⟦F,y,y,z⟧)) (λ F . (⟦F,y,z,y⟧))
      (λ F . (⟦F,z,y,y⟧))
(* three place (y,z,z) *) (λ F . (⟦F,y,z,z⟧)) (λ F . (⟦F,z,y,z⟧))
      (λ F . (⟦F,z,z,y⟧))
(* three place (y,y,y) *) (λ F . (⟦F,y,y,y⟧))
(* three place (z,z,z) *) (λ F . (⟦F,z,z,z⟧))
(* x y z *)
(* three place (x,...) *) (λ F . (⟦F,x,y,z⟧)) (λ F . (⟦F,x,z,y⟧))
(* three place (y,...) *) (λ F . (⟦F,y,x,z⟧)) (λ F . (⟦F,y,z,x⟧))
(* three place (z,...) *) (λ F . (⟦F,z,x,y⟧)) (λ F . (⟦F,z,y,x⟧))
unfolding IsProperInXYZ-def
by (auto simp: meta-defs meta-aux)

```

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

2.3 Validity Syntax

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

3 General Quantification

Remark 8. In order to define general quantifiers that can act on individuals as well as relations a type class is introduced which assumes the semantics of the all quantifier. This type class is then instantiated for individuals and relations.

3.1 Type Class

```

class quantifiable = fixes forall :: ('a  $\Rightarrow$  o)  $\Rightarrow$  o (binder  $\forall$  [8] 9)
  assumes quantifiable-T8: ( $w \models (\forall \ x . \psi \ x)$ ) = ( $\forall \ x . (w \models (\psi \ x))$ )
begin
end

```

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

3.2 Instantiations

instantiation $\nu :: \text{quantifiable}$

begin

definition *forall- ν* :: $(\nu \Rightarrow o) \Rightarrow o$ **where** *forall- ν* \equiv *forall $_{\nu}$*

instance proof

fix $w :: i$ **and** $\psi :: \nu \Rightarrow o$

show $(w \models \forall x . \psi x) = (\forall x . (w \models \psi x))$

unfolding *forall- ν -def* **using** *Semantics.T8- ν* .

qed

end

instantiation $o :: \text{quantifiable}$

begin

definition *forall-o* :: $(o \Rightarrow o) \Rightarrow o$ **where** *forall-o* \equiv *forall $_o$*

instance proof

fix $w :: i$ **and** $\psi :: o \Rightarrow o$

show $(w \models \forall x . \psi x) = (\forall x . (w \models \psi x))$

unfolding *forall-o-def* **using** *Semantics.T8-o* .

qed

end

instantiation $\Pi_1 :: \text{quantifiable}$

begin

definition *forall- Π_1* :: $(\Pi_1 \Rightarrow o) \Rightarrow o$ **where** *forall- Π_1* \equiv *forall $_1$*

instance proof

fix $w :: i$ **and** $\psi :: \Pi_1 \Rightarrow o$

show $(w \models \forall x . \psi x) = (\forall x . (w \models \psi x))$

unfolding *forall- Π_1 -def* **using** *Semantics.T8-1* .

qed

end

instantiation $\Pi_2 :: \text{quantifiable}$

begin

definition *forall- Π_2* :: $(\Pi_2 \Rightarrow o) \Rightarrow o$ **where** *forall- Π_2* \equiv *forall $_2$*

instance proof

fix $w :: i$ **and** $\psi :: \Pi_2 \Rightarrow o$

show $(w \models \forall x . \psi x) = (\forall x . (w \models \psi x))$

unfolding *forall- Π_2 -def* **using** *Semantics.T8-2* .

qed

end

instantiation $\Pi_3 :: \text{quantifiable}$

begin

definition *forall- Π_3* :: $(\Pi_3 \Rightarrow o) \Rightarrow o$ **where** *forall- Π_3* \equiv *forall $_3$*

instance proof

fix $w :: i$ **and** $\psi :: \Pi_3 \Rightarrow o$

show $(w \models \forall x . \psi x) = (\forall x . (w \models \psi x))$

unfolding *forall- Π_3 -def* **using** *Semantics.T8-3* .

qed

end

4 Basic Definitions

4.1 Derived Connectives

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

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

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

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

4.2 Abstract and Ordinary Objects

definition $Ordinary :: \Pi_1 (O!)$ **where** $Ordinary \equiv \lambda x . \Diamond (E!, x^P)$
definition $Abstract :: \Pi_1 (A!)$ **where** $Abstract \equiv \lambda x . \neg \Diamond (E!, x^P)$

4.3 Identity Definitions

definition $basic-identity_E :: \Pi_2$ **where**
 $basic-identity_E \equiv \lambda^2 (\lambda x y . (O!, x^P) \ \&\ (O!, y^P) \ \&\ \Box (\forall F . (F, x^P) \equiv (F, y^P)))$

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

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

definition $basic-identity_1$ (**infixl** $=_1$ 63) **where**
 $basic-identity_1 \equiv \lambda F G . \Box (\forall x . (x^P, F) \equiv (x^P, G))$

definition $basic-identity_2 :: \Pi_2 \Rightarrow \Pi_2 \Rightarrow o$ (**infixl** $=_2$ 63) **where**
 $basic-identity_2 \equiv \lambda F G . \forall x . ((\lambda y . (F, x^P, y^P)) =_1 (\lambda y . (G, x^P, y^P))) \ \&\ ((\lambda y . (F, y^P, x^P)) =_1 (\lambda y . (G, y^P, x^P)))$

definition $basic-identity_3 :: \Pi_3 \Rightarrow \Pi_3 \Rightarrow o$ (**infixl** $=_3$ 63) **where**
 $basic-identity_3 \equiv \lambda F G . \forall x y . (\lambda z . (F, z^P, x^P, y^P)) =_1 (\lambda z . (G, z^P, x^P, y^P)) \ \&\ (\lambda z . (F, x^P, z^P, y^P)) =_1 (\lambda z . (G, x^P, z^P, y^P)) \ \&\ (\lambda z . (F, x^P, y^P, z^P)) =_1 (\lambda z . (G, x^P, y^P, z^P))$

definition $basic-identity_0 :: o \Rightarrow o \Rightarrow o$ (**infixl** $=_0$ 63) **where**
 $basic-identity_0 \equiv \lambda F G . (\lambda y . F) =_1 (\lambda y . G)$

5 MetaSolver

Remark 9. *meta-solver is a resolution prover that translates expressions in the embedded logic to expressions in the meta-logic, resp. semantic expressions. The rules for connectives, quantifiers, exemplification and encoding are straightforward. Furthermore, rules for the defined identities are derived. The defined identities in the embedded logic coincide with the meta-logical equality.*

locale *MetaSolver*
begin
interpretation *Semantics* .

named-theorems *meta-intro*

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

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

5.1 Rules for Implication

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

5.2 Rules for Negation

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

5.3 Rules for Conjunction

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

5.4 Rules for Equivalence

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

5.5 Rules for Disjunction

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

5.6 Rules for Necessity

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

5.7 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*)

5.8 Rules for Quantification

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

5.8.1 Rules for Existence

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

5.9 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*)

5.10 Rules for Encoding

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

5.11 Rules for Exemplification

5.11.1 Zero-place Relations

lemma *ExeOI*[*meta-intro*]:
assumes $\exists r. \text{Some } r = d_0 \ p \wedge ex0 \ r \ v$
shows $[\llbracket p \rrbracket \text{ in } v]$
using *assms* **by** (*auto simp: Semantics.T3*)

lemma *Exe0E*[*meta-elim*]:
assumes $[(\downarrow p)]$ *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*]:
 $[(\downarrow p)]$ *in* $v = (\exists r . \text{Some } r = d_0 p \wedge \text{ex0 } r v)$
by (*auto simp: Semantics.T3*)

5.11.2 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 $[(\downarrow F, x)]$ *in* v
using *assms* **by** (*auto simp: Semantics.T1-1*)
lemma *Exe1E*[*meta-elim*]:
assumes $[(\downarrow F, x)]$ *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*]:
 $[(\downarrow F, x)]$ *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*)

5.11.3 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 $[(\downarrow F, x, y)]$ *in* v
using *assms* **by** (*auto simp: Semantics.T1-2*)
lemma *Exe2E*[*meta-elim*]:
assumes $[(\downarrow F, x, y)]$ *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*]:
 $[(\downarrow F, x, y)]$ *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*)

5.11.4 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 $[(\downarrow F, x, y, z)]$ *in* v
using *assms* **by** (*auto simp: Semantics.T1-3*)
lemma *Exe3E*[*meta-elim*]:
assumes $[(\downarrow F, x, y, z)]$ *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*]:
 $[(\downarrow F, x, y, z)]$ *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*)

5.12 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 $[(\downarrow O!, x)]$ *in* v

```

proof –
  have IsProperInX ( $\lambda x. \Diamond \langle E!, x \rangle$ )
    by show-proper
  moreover have  $\langle \Diamond \langle E!, x \rangle \text{ in } v$ 
    apply meta-solver
    using ConcretenessSemantics1 proper1 assms by fast
  ultimately show  $\langle \langle O!, x \rangle \text{ in } v$ 
    unfolding Ordinary-def
    using D5-1 proper1 assms ConcretenessSemantics1 Exe1S
    by blast
qed
lemma OrdE[meta-elim]:
  assumes  $\langle \langle O!, x \rangle \text{ in } v$ 
  shows  $\exists o_1 y. \text{Some } o_1 = d_\kappa x \wedge o_1 = \omega\nu y$ 
proof –
  have  $\exists r o_1. \text{Some } r = d_1 O! \wedge \text{Some } o_1 = d_\kappa x \wedge o_1 \in \text{exl } r v$ 
    using assms Exe1E by simp
  moreover have IsProperInX ( $\lambda x. \Diamond \langle E!, x \rangle$ )
    by show-proper
  ultimately have  $\langle \Diamond \langle E!, x \rangle \text{ in } v$ 
    using D5-1 unfolding Ordinary-def by fast
  thus ?thesis
    apply – apply meta-solver
    using ConcretenessSemantics2 by blast
qed
lemma OrdS[meta-cong]:
   $\langle \langle O!, x \rangle \text{ in } v \rangle = (\exists o_1 y. \text{Some } o_1 = d_\kappa x \wedge o_1 = \omega\nu y)$ 
using OrdI OrdE by blast

```

5.13 Rules for Being Abstract

```

lemma AbsI[meta-intro]:
  assumes  $\exists o_1 y. \text{Some } o_1 = d_\kappa x \wedge o_1 = \alpha\nu y$ 
  shows  $\langle \langle A!, x \rangle \text{ in } v$ 
proof –
  have IsProperInX ( $\lambda x. \neg \Diamond \langle E!, x \rangle$ )
    by show-proper
  moreover have  $\langle \neg \Diamond \langle E!, x \rangle \text{ in } v$ 
    apply meta-solver
    using ConcretenessSemantics2 proper1 assms
    by (metis  $\nu.\text{distinct}(1)$  option.sel)
  ultimately show  $\langle \langle A!, x \rangle \text{ in } v$ 
    unfolding Abstract-def
    using D5-1 proper1 assms ConcretenessSemantics1 Exe1S
    by blast
qed
lemma AbsE[meta-elim]:
  assumes  $\langle \langle A!, x \rangle \text{ in } v$ 
  shows  $\exists o_1 y. \text{Some } o_1 = d_\kappa x \wedge o_1 = \alpha\nu y$ 
proof –
  have 1: IsProperInX ( $\lambda x. \neg \Diamond \langle E!, x \rangle$ )
    by show-proper
  have  $\exists r o_1. \text{Some } r = d_1 A! \wedge \text{Some } o_1 = d_\kappa x \wedge o_1 \in \text{exl } r v$ 
    using assms Exe1E by simp
  moreover hence  $\langle \neg \Diamond \langle E!, x \rangle \text{ in } v$ 
    using D5-1[OF 1]
    unfolding Abstract-def by fast
  ultimately show ?thesis
    apply – apply meta-solver
    using ConcretenessSemantics1 proper1
    by (metis  $\nu.\text{exhaust}$ )
qed
lemma AbsS[meta-cong]:

```

$[(\lambda!x) \text{ in } v] = (\exists o_1 y. \text{Some } o_1 = d_\kappa x \wedge o_1 = \alpha\nu y)$
using *AbsI AbsE* **by** *blast*

5.14 Rules for Definite Descriptions

lemma *TheEqI*:
assumes $\bigwedge x. [\varphi x \text{ in } dw] = [\psi x \text{ in } dw]$
shows $(\iota x. \varphi x) = (\iota x. \psi x)$
proof –
have $1: d_\kappa (\iota x. \varphi x) = d_\kappa (\iota x. \psi x)$
using *assms D3 unfolding w0-def* **by** *simp*
{
assume $\exists o_1. \text{Some } o_1 = d_\kappa (\iota x. \varphi x)$
hence *?thesis* **using** $1 d_\kappa\text{-inject}$ **by** *force*
}
moreover **{**
assume $\neg(\exists o_1. \text{Some } o_1 = d_\kappa (\iota x. \varphi x))$
hence *?thesis* **using** $1 D3$
by (*metis d_κ.rep-eq evalκ-inverse*)
}
ultimately show *?thesis* **by** *blast*
qed

5.15 Rules for Identity

5.15.1 Ordinary Objects

lemma *EqEI[meta-intro]*:
assumes $\exists o_1 o_2. \text{Some } (\omega\nu o_1) = d_\kappa x \wedge \text{Some } (\omega\nu o_2) = d_\kappa y \wedge o_1 = o_2$
shows $[x =_E y \text{ in } v]$
proof –
obtain $o_1 o_2$ **where** $1:$
 $\text{Some } (\omega\nu o_1) = d_\kappa x \wedge \text{Some } (\omega\nu o_2) = d_\kappa y \wedge o_1 = o_2$
using *assms* **by** *auto*
obtain r **where** $2:$
 $\text{Some } r = d_2 \text{ basic-identity}_E$
using *properx₂* **by** *auto*
have $[(\lambda O!,x) \ \& \ (\lambda O!,y) \ \& \ \Box(\forall F. (\lambda F,x) \equiv (\lambda F,y))] \text{ in } v]$
proof –
have $[(\lambda O!,x) \text{ in } v] \wedge [(\lambda O!,y) \text{ in } v]$
using *OrdI 1* **by** *blast*
moreover have $[\Box(\forall F. (\lambda F,x) \equiv (\lambda F,y))] \text{ in } v]$
apply *meta-solver* **using** 1 **by** *force*
ultimately show *?thesis* **using** *ConjI* **by** *simp*
qed
moreover have *IsProperInXY* $(\lambda x y. (\lambda O!,x) \ \& \ (\lambda O!,y) \ \& \ \Box(\forall F. (\lambda F,x) \equiv (\lambda F,y)))$
by *show-proper*
ultimately have $(\omega\nu o_1, \omega\nu o_2) \in \text{ex2 } r v$
using *D5-2 1 2*
unfolding *basic-identity_E-def* **by** *fast*
thus $[x =_E y \text{ in } v]$
using *Exe2I 1 2*
unfolding *basic-identity_E-infix-def basic-identity_E-def*
by *blast*
qed
lemma *EqEE[meta-elim]*:
assumes $[x =_E y \text{ in } v]$
shows $\exists o_1 o_2. \text{Some } (\omega\nu o_1) = d_\kappa x \wedge \text{Some } (\omega\nu o_2) = d_\kappa y \wedge o_1 = o_2$
proof –
have *IsProperInXY* $(\lambda x y. (\lambda O!,x) \ \& \ (\lambda O!,y) \ \& \ \Box(\forall F. (\lambda F,x) \equiv (\lambda F,y)))$
by *show-proper*
hence $1: [(\lambda O!,x) \ \& \ (\lambda O!,y) \ \& \ \Box(\forall F. (\lambda F,x) \equiv (\lambda F,y))] \text{ in } v]$

```

    using assms unfolding basic-identityE-def basic-identityE-infix-def
    using D4-2 T1-2 D5-2 by meson
  hence 2:  $\exists o_1 o_2. \text{Some } (\omega\nu o_1) = d_\kappa x$ 
              $\wedge \text{Some } (\omega\nu o_2) = d_\kappa y$ 
    apply (subst (asm) ConjS)
    apply (subst (asm) ConjS)
    using OrdE by auto
  then obtain  $o_1 o_2$  where 3:
    Some  $(\omega\nu o_1) = d_\kappa x \wedge \text{Some } (\omega\nu o_2) = d_\kappa y$ 
    by auto
  have  $\exists r. \text{Some } r = d_1 (\lambda z. \text{makeo } (\lambda w s. d_\kappa (z^P) = \text{Some } (\omega\nu o_1)))$ 
    using properx1 by auto
  then obtain  $r$  where 4:
    Some  $r = d_1 (\lambda z. \text{makeo } (\lambda w s. d_\kappa (z^P) = \text{Some } (\omega\nu o_1)))$ 
    by auto
  hence 5:  $r = (\lambda u s w. \exists x. \nu\nu x = u \wedge \text{Some } x = \text{Some } (\omega\nu o_1))$ 
    unfolding lambdabinder1-def d1-def dκ-proper
    apply transfer
    by simp
  have  $[\Box(\forall F. \langle F, x \rangle \equiv \langle F, y \rangle)] \text{ in } v$ 
    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 AllE by fast
  hence  $\forall v. ((\omega\nu o_1) \in \text{ex1 } r v) = ((\omega\nu o_2) \in \text{ex1 } r v)$ 
    using 2 4 unfolding valid-in-def
    by (metis 3 6 d1.rep-eq dκ-inject dκ-proper ex1-def evalo-inverse exe1.rep-eq
        mem-Collect-eq option.sel rep-proper-id νκ-proper valid-in.abs-eq)
  moreover have  $(\omega\nu o_1) \in \text{ex1 } r v$ 
    unfolding 5 ex1-def by simp
  ultimately have  $(\omega\nu o_2) \in \text{ex1 } r v$ 
    by auto
  hence  $o_1 = o_2$  unfolding 5 ex1-def by (auto simp: meta-aux)
  thus ?thesis
    using 3 by auto
qed
lemma EqES[meta-subst]:
 $[x =_E y \text{ in } v] = (\exists o_1 o_2. \text{Some } (\omega\nu o_1) = d_\kappa x \wedge \text{Some } (\omega\nu o_2) = d_\kappa y$ 
 $\wedge o_1 = o_2)$ 
  using EqEI EqEE by blast

```

5.15.2 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  $[x =_E y \vee \langle A!, x \rangle \ \& \ \langle A!, y \rangle \ \& \ \Box(\forall F. \langle x, F \rangle \equiv \langle y, F \rangle)] \text{ in } v$ 
    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 1:  $[(\lambda A!x) \ \& \ (\lambda A!y) \ \& \ \Box(\forall F. \llbracket x, F \rrbracket \equiv \llbracket y, F \rrbracket)]$  in  $v$ 
  hence 2:  $(\exists o_1 o_2 X Y. \text{Some } o_1 = d_\kappa x \wedge \text{Some } o_2 = d_\kappa y$ 
     $\wedge o_1 = \alpha\nu X \wedge o_2 = \alpha\nu Y)$ 
    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:  $[\Box(\forall F. \llbracket x, F \rrbracket \equiv \llbracket y, F \rrbracket)]$  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 AllE EquivE by fast
  hence 7:  $\forall v r. (\exists o_1. \text{Some } o_1 = d_\kappa x \wedge o_1 \in \text{en } r)$ 
     $= (\exists o_1. \text{Some } o_1 = d_\kappa y \wedge o_1 \in \text{en } r)$ 
    apply – apply meta-solver
    using propex1 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 dκ-inject dκ-proper apply simp
    by (metis option.inject)
  hence  $\forall r. (o_1 \in r) = (o_2 \in r)$ 
    unfolding en-def using 3
    by (metis Collect-cong Collect-mem-eq ν.simps(6)
      mem-Collect-eq makeΠ1-cases)
  hence  $(o_1 \in \{x \mid o_1 = x\}) = (o_2 \in \{x \mid o_1 = x\})$ 
    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.dκ-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  $\varphi$  where  $\varphi\text{-prop}$ :
     $\varphi = (\lambda \alpha. \text{makeo } (\lambda w s. (\exists o_1 o_2. \text{Some } o_1 = d_\kappa x$ 
       $\wedge \text{Some } o_2 = d_\kappa \alpha \wedge o_1 = o_2)))$ 
    by auto
  ultimately have  $(v \models \varphi x) = (v \models \varphi y)$  by metis
  moreover have  $(v \models \varphi x)$ 
    using assms unfolding  $\varphi\text{-prop}$  basic-identityκ-def
    by (metis (mono-tags, lifting) AbsS ConjE DisjS
      EqES valid-in.abs-eq)
  ultimately have  $(v \models \varphi y)$  by auto
  thus ?thesis
    unfolding  $\varphi\text{-prop}$ 
    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 EqκI EqκE by blast

```

5.15.3 One-Place Relations

```

lemma Eq1I[meta-intro]:  $F = G \implies [F =_1 G \text{ in } v]$ 
  unfolding basic-identity1-def

```

```

apply (rule BoxI, rule AllI, rule EquivI)
by simp
lemma Eq1E[meta-elim]:  $[F =_1 G \text{ in } v] \implies F = G$ 
unfolding basic-identity1-def
apply (drule BoxE, drule-tac  $x=(\alpha v \{ F \})$  in AllE, drule EquivE)
apply (simp add: Semantics.T2)
unfolding en-def dκ-def d1-def
using νκ-proper rep-proper-id
by (simp add: rep-def proper-def meta-aux νκ.rep-eq)
lemma Eq1S[meta-subst]:  $[F =_1 G \text{ in } v] = (F = G)$ 
using Eq1I Eq1E by auto
lemma Eq1-prop:  $[F =_1 G \text{ in } v] \implies [\varphi F \text{ in } v] = [\varphi G \text{ in } v]$ 
using Eq1E by blast

```

5.15.4 Two-Place Relations

```

lemma Eq2I[meta-intro]:  $F = G \implies [F =_2 G \text{ in } v]$ 
unfolding basic-identity2-def
apply (rule AllI, rule ConjI, (subst Eq1S)+)
by simp
lemma Eq2E[meta-elim]:  $[F =_2 G \text{ in } v] \implies F = G$ 
proof –
  assume  $[F =_2 G \text{ in } v]$ 
  hence 1:  $[\forall x. (\lambda y. \langle F, x^P, y^P \rangle) =_1 (\lambda y. \langle G, x^P, y^P \rangle) \text{ in } v]$ 
  unfolding basic-identity2-def
  apply – apply meta-solver by auto
  {
    fix u v s w
    obtain x where x-def:  $\nu v \ x = v$  by (metis νv-surj surj-def)
    obtain a where a-def:
       $a = (\lambda u \ s \ w. \exists x a. \nu v \ x a = u \wedge \text{eval}\Pi_2 \ F \ (\nu v \ x) \ (\nu v \ x a) \ s \ w)$ 
      by auto
    obtain b where b-def:
       $b = (\lambda u \ s \ w. \exists x a. \nu v \ x a = u \wedge \text{eval}\Pi_2 \ G \ (\nu v \ x) \ (\nu v \ x a) \ s \ w)$ 
      by auto
    have a = b unfolding a-def b-def
      using 1 apply – apply meta-solver
      by (auto simp: meta-defs meta-aux makeΠ1-inject)
    hence a u s w = b u s w by auto
    hence (evalΠ2 F (νv x) u s w) = (evalΠ2 G (νv x) u s w)
      unfolding a-def b-def
      by (metis (no-types, hide-lams) νv-surj surj-def)
    hence (evalΠ2 F v u s w) = (evalΠ2 G v u s w)
      unfolding x-def by auto
  }
  hence (evalΠ2 F) = (evalΠ2 G) by blast
  thus F = G by (simp add: evalΠ2-inject)
qed
lemma Eq2S[meta-subst]:  $[F =_2 G \text{ in } v] = (F = G)$ 
using Eq2I Eq2E by auto
lemma Eq2-prop:  $[F =_2 G \text{ in } v] \implies [\varphi F \text{ in } v] = [\varphi G \text{ in } v]$ 
using Eq2E by blast

```

5.15.5 Three-Place Relations

```

lemma Eq3I[meta-intro]:  $F = G \implies [F =_3 G \text{ in } v]$ 
apply (simp add: meta-defs meta-aux conn-defs forall-ν-def basic-identity3-def)
using MetaSolver.Eq1I valid-in.rep-eq by auto
lemma Eq3E[meta-elim]:  $[F =_3 G \text{ in } v] \implies F = G$ 
proof –

```

```

  assume  $[F =_3 G \text{ in } v]$ 
  hence 1:  $[\forall x \ y. (\lambda z. \langle F, x^P, y^P, z^P \rangle) =_1 (\lambda z. \langle G, x^P, y^P, z^P \rangle) \text{ in } v]$ 

```

```

unfolding basic-identity3-def
apply – apply meta-solver by auto
{
  fix u v r s w
  obtain x where x-def:  $\nu v\ x = v$  by (metis  $\nu v$ -surj surj-def)
  obtain y where y-def:  $\nu v\ y = r$  by (metis  $\nu v$ -surj surj-def)
  obtain a where a-def:
     $a = (\lambda u\ s\ w. \exists x a. \nu v\ xa = u \wedge \text{eval}\Pi_3\ F\ (\nu v\ x)\ (\nu v\ y)\ (\nu v\ xa)\ s\ w)$ 
    by auto
  obtain b where b-def:
     $b = (\lambda u\ s\ w. \exists x a. \nu v\ xa = u \wedge \text{eval}\Pi_3\ G\ (\nu v\ x)\ (\nu v\ y)\ (\nu v\ xa)\ s\ w)$ 
    by auto
  have  $a = b$  unfolding a-def b-def
    using 1 apply – apply meta-solver
    by (auto simp: meta-defs meta-aux make $\Pi_1$ -inject)
  hence  $a\ u\ s\ w = b\ u\ s\ w$  by auto
  hence  $(\text{eval}\Pi_3\ F\ (\nu v\ x)\ (\nu v\ y)\ u\ s\ w) = (\text{eval}\Pi_3\ G\ (\nu v\ x)\ (\nu v\ y)\ u\ s\ w)$ 
    unfolding a-def b-def
    by (metis (no-types, hide-lams)  $\nu v$ -surj surj-def)
  hence  $(\text{eval}\Pi_3\ F\ v\ r\ u\ s\ w) = (\text{eval}\Pi_3\ G\ v\ r\ u\ s\ w)$ 
    unfolding x-def y-def by auto
}
hence  $(\text{eval}\Pi_3\ F) = (\text{eval}\Pi_3\ G)$  by blast
thus  $F = G$  by (simp add: eval $\Pi_3$ -inject)
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

```

5.15.6 Propositions

```

lemma Eq0I[meta-intro]:  $x = y \implies [x =_0\ y\ \text{in}\ v]$ 
unfolding basic-identity0-def by (simp add: Eq1S)
lemma Eq0E[meta-elim]:  $[F =_0\ G\ \text{in}\ v] \implies F = G$ 
proof –
  assume  $[F =_0\ G\ \text{in}\ v]$ 
  hence  $[(\lambda y. F) =_1\ (\lambda y. G)\ \text{in}\ v]$ 
    unfolding basic-identity0-def by simp
  hence  $(\lambda y. F) = (\lambda y. G)$ 
    using Eq1S by simp
  hence  $(\lambda u\ s\ w. (\exists x. \nu v\ x = u) \wedge \text{evalo}\ F\ s\ w)$ 
     $= (\lambda u\ s\ w. (\exists x. \nu v\ x = u) \wedge \text{evalo}\ G\ s\ w)$ 
    apply (simp add: meta-defs meta-aux)
    by (metis (no-types, lifting) UNIV-I make $\Pi_1$ -inverse)
  hence  $\bigwedge s\ w. (\text{evalo}\ F\ s\ w) = (\text{evalo}\ G\ s\ w)$ 
    by metis
  hence  $(\text{evalo}\ F) = (\text{evalo}\ G)$  by blast
  thus  $F = G$ 
    by (metis evalo-inverse)
qed
lemma Eq0S[meta-subst]:  $[F =_0\ G\ \text{in}\ v] = (F = G)$ 
using Eq0I Eq0E by auto
lemma Eq0-prop:  $[F =_0\ G\ \text{in}\ v] \implies [\varphi\ F\ \text{in}\ v] = [\varphi\ G\ \text{in}\ v]$ 
using Eq0E by blast

```

end

6 General Identity

Remark 10. *In order to define a general identity symbol that can act on all types of terms a type class is introduced which assumes the substitution property which is needed to derive the corresponding axiom. This type class is instantiated for all relation types, individual terms and individuals.*

6.1 Type Classes

```

class identifiable =
fixes identity :: 'a ⇒ 'a ⇒ o (infixl = 63)
assumes l-identity:
  w ⊢ x = y ⇒ w ⊢ φ x ⇒ w ⊢ φ y
begin
  abbreviation notequal (infixl ≠ 63) where
    notequal ≡ λ x y . ¬(x = y)
end

class quantifiable-and-identifiable = quantifiable + identifiable
begin
  definition exists-unique::('a ⇒ o) ⇒ o (binder ∃! [8] 9) where
    exists-unique ≡ λ φ . ∃ α . φ α & (∀ β. φ β → β = α)

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

```

6.2 Instantiations

```

instantiation κ :: identifiable
begin
  definition identity-κ where identity-κ ≡ basic-identity_κ
  instance proof
    fix x y :: κ and w φ
    show [x = y in w] ⇒ [φ x in w] ⇒ [φ y in w]
      unfolding identity-κ-def
      using MetaSolver.Eqκ-prop ..
    qed
  end

instantiation ν :: identifiable
begin
  definition identity-ν where identity-ν ≡ λ x y . xP = yP
  instance proof
    fix α :: ν and β :: ν and v φ
    assume v ⊢ α = β
    hence v ⊢ αP = βP
      unfolding identity-ν-def by auto
    hence ∧φ.(v ⊢ φ (αP)) ⇒ (v ⊢ φ (βP))
      using l-identity by auto
    hence (v ⊢ φ (rep (αP))) ⇒ (v ⊢ φ (rep (βP)))
      by meson
    thus (v ⊢ φ α) ⇒ (v ⊢ φ β)
      by (simp only: rep-proper-id)
    qed
  end

instantiation Π1 :: identifiable
begin
  definition identity-Π1 where identity-Π1 ≡ basic-identity1
  instance proof
    fix F G :: Π1 and w φ

```



```

show (w ⊨ F = G) ⇒ (w ⊨ φ F) ⇒ (w ⊨ φ G)
  unfolding identity-Π1-def using MetaSolver.Eq1-prop ..
qed
end

```

```

instantiation Π2 :: identifiable
begin
  definition identity-Π2 where identity-Π2 ≡ basic-identity2
  instance proof
    fix F G :: Π2 and w φ
    show (w ⊨ F = G) ⇒ (w ⊨ φ F) ⇒ (w ⊨ φ G)
      unfolding identity-Π2-def using MetaSolver.Eq2-prop ..
    qed
  end
end

```

```

instantiation Π3 :: identifiable
begin
  definition identity-Π3 where identity-Π3 ≡ basic-identity3
  instance proof
    fix F G :: Π3 and w φ
    show (w ⊨ F = G) ⇒ (w ⊨ φ F) ⇒ (w ⊨ φ G)
      unfolding identity-Π3-def using MetaSolver.Eq3-prop ..
    qed
  end
end

```

```

instantiation o :: identifiable
begin
  definition identity-o where identity-o ≡ basic-identity0
  instance proof
    fix F G :: o and w φ
    show (w ⊨ F = G) ⇒ (w ⊨ φ F) ⇒ (w ⊨ φ G)
      unfolding identity-o-def using MetaSolver.Eq0-prop ..
    qed
  end
end

```

```

instance ν :: quantifiable-and-identifiable ..
instance Π1 :: quantifiable-and-identifiable ..
instance Π2 :: quantifiable-and-identifiable ..
instance Π3 :: quantifiable-and-identifiable ..
instance o :: quantifiable-and-identifiable ..

```

6.3 New Identity Definitions

Remark 11. *The basic definitions of identity use type specific quantifiers and identity symbols. Equivalent definitions that use the general identity symbol and general quantifiers are provided.*

```

named-theorems identity-defs
lemma identityE-def[identity-defs]:
  basic-identityE ≡ λ2 (λx y. (O!, xP) & (O!, yP) & □(∀ F. (F, xP) ≡ (F, yP)))
  unfolding basic-identityE-def forall-Π1-def by simp
lemma identityE-infix-def[identity-defs]:
  x =E y ≡ (basic-identityE, x, y) using basic-identityE-infix-def .
lemma identityκ-def[identity-defs]:
  op = ≡ λx y. x =E y ∨ (A!, x) & (A!, y) & □(∀ F. (x, F) ≡ (y, F))
  unfolding identity-κ-def basic-identityκ-def forall-Π1-def by simp
lemma identityν-def[identity-defs]:
  op = ≡ λx y. (xP) =E (yP) ∨ (A!, xP) & (A!, yP) & □(∀ F. (xP, F) ≡ (yP, F))
  unfolding identity-ν-def identityκ-def by simp
lemma identity1-def[identity-defs]:
  op = ≡ λF G. □(∀ x. (xP, F) ≡ (xP, G))
  unfolding identity-Π1-def basic-identity1-def forall-ν-def by simp
lemma identity2-def[identity-defs]:

```

```

op = ≡ λF G. ∀ x. (λy. (F, xP, yP)) = (λy. (G, xP, yP))
      & (λy. (F, yP, xP)) = (λy. (G, yP, xP))
unfolding identity-Π2-def identity-Π1-def basic-identity2-def forall-ν-def by simp
lemma identity3-def[identity-defs]:
op = ≡ λF G. ∀ x y. (λz. (F, zP, xP, yP)) = (λz. (G, zP, xP, yP))
      & (λz. (F, xP, zP, yP)) = (λz. (G, xP, zP, yP))
      & (λz. (F, xP, yP, zP)) = (λz. (G, xP, yP, zP))
unfolding identity-Π3-def identity-Π1-def basic-identity3-def forall-ν-def by simp
lemma identityo-def[identity-defs]: op = ≡ λF G. (λy. F) = (λy. G)
unfolding identity-o-def identity-Π1-def basic-identity0-def by simp

```

7 The Axioms of PLM

Remark 12. *The axioms of PLM can now be derived from the Semantics and the model structure.*

```

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

```

Remark 13. *The special syntax $[[\cdot]]$ is introduced for stating the axioms. Modally-fragile axioms are stated with the syntax for actual validity $[\cdot]$.*

```

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

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

```

7.1 Closures

Remark 14. *Rules resembling the concepts of closures in PLM are derived. Theorem attributes are introduced to aid in the instantiation of the axioms.*

```

lemma axiom-instance[axiom]: [[φ]] ⇒ [φ in v]
  unfolding axiom-def by simp
lemma closures-universal[axiom]: (Λx. [[φ x]]) ⇒ [[∀ x. φ x]]
  by axiom-meta-solver
lemma closures-actualization[axiom]: [[φ]] ⇒ [[A φ]]
  by axiom-meta-solver
lemma closures-necessitation[axiom]: [[φ]] ⇒ [[□ φ]]
  by axiom-meta-solver
lemma necessitation-averse-axiom-instance[axiom]: [φ] ⇒ [φ in dw]
  by axiom-meta-solver
lemma necessitation-averse-closures-universal[axiom]: (Λx. [φ x]) ⇒ [∀ x. φ x]
  by axiom-meta-solver

```

```

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

```

```

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

```

```

attribute-setup axiom-necessitation = ⟨⟨

```

```

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

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

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

```

7.2 Axioms for Negations and Conditionals

```

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

```

7.3 Axioms of Identity

```

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

```

7.4 Axioms of Quantification

```

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

```

inductive SimpleExOrEnc

```

where SimpleExOrEnc ( $\lambda x . \langle F, x \rangle$ )
  | SimpleExOrEnc ( $\lambda x . \langle F, x, y \rangle$ )
  | SimpleExOrEnc ( $\lambda x . \langle F, y, x \rangle$ )
  | SimpleExOrEnc ( $\lambda x . \langle F, x, y, z \rangle$ )
  | SimpleExOrEnc ( $\lambda x . \langle F, y, x, z \rangle$ )
  | SimpleExOrEnc ( $\lambda x . \langle F, y, z, x \rangle$ )
  | SimpleExOrEnc ( $\lambda x . \langle x, F \rangle$ )

```

```

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

```

```

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

```

7.5 Axioms of Actuality

```

lemma logic-actual[axiom]:  $[(\mathcal{A}\varphi) \equiv \varphi]$ 
  by axiom-meta-solver
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*]:
 $[[\Diamond(\exists x. (\Box E!, x^P)) \ \& \ \Diamond \neg(\Box E!, x^P)) \ \& \ \Diamond \neg(\exists x. (\Box E!, x^P) \ \& \ \Diamond \neg(\Box E!, x^P))]]$
 unfolding *axiom-def*
 using *PossiblyContingentObjectExistsAxiom*
 PossiblyNoContingentObjectExistsAxiom
 apply (simp add: meta-defs meta-aux conn-defs forall- ν -def
 split: ν .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 $[x^P = (\iota x. \varphi x) \text{ in } v]$
 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 – unfolding *identity- κ -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 $[\mathcal{A}\varphi z \text{ in } v] \Longrightarrow [(z^P) = (x^P) \text{ in } v]$
 unfolding *identity- κ -def* apply *meta-solver*
 using 4 5 2 *d κ -proper w_0 -def* by *auto*
 moreover have $[(z^P) = (x^P) \text{ in } v] \Longrightarrow [\mathcal{A}\varphi z \text{ in } v]$
 unfolding *identity- κ -def* apply *meta-solver*
 using 2 4 5
 by (simp add: *d κ -proper 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- ν -def*
 by (simp add: *AllI EquivS*)
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 – **unfolding** *identity- ν -def identity- κ -def* by *meta-solver*
 hence $\forall z. (dw \models \varphi z) = (z = x)$
 by (*simp add: d κ -proper*)
 moreover hence $x = (THE z. (dw \models \varphi z))$ by *simp*
 ultimately have $x^P = (\iota x. \varphi x)$
 using *D3 d κ -inject d κ -proper w₀-def* by *presburger*
 thus $[x^P = (\iota x. \varphi x)]$ in *v*
 using *Eq κ S unfolding identity- κ -def* by (*metis d κ -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 *IsProperInX* φ
 shows $[(\lambda x. \varphi (x^P), x^P) \equiv \varphi (x^P)]$
 apply *axiom-meta-solver*
 using *D5-1[OF assms] d κ -proper proper_{x1}*
 by *metis*

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

assumes *IsProperInXY* 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] d κ -proper proper_{x2}*
 by *metis*

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

assumes *IsProperInXYZ* 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 $[(\lambda^3 (\lambda x y z. \varphi (x^P) (y^P) (z^P))), x^P, y^P, z^P] \rightarrow \varphi (x^P) (y^P) (z^P)]$
 apply *axiom-meta-solver* using *D5-3[OF assms]* by *auto*
 moreover have
 $[(\varphi (x^P) (y^P) (z^P) \rightarrow (\lambda^3 (\lambda x y z. \varphi (x^P) (y^P) (z^P))), x^P, y^P, z^P]$
 apply *axiom-meta-solver*
 using *D5-3[OF assms] d κ -proper proper_{x3}*
 by (*metis (no-types, lifting)*)
 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*]:

$[(\lambda x. (F, x^P)) = F]$
unfolding *axiom-def*
 apply (*rule allI*)
unfolding *identity- Π_1 -def* apply (*rule Eq₁I*)
 using *D4-1 d₁-inject* by *simp*

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

$[(\lambda^2 (\lambda x y. (F, x^P, y^P))) = F]$
unfolding *axiom-def*
 apply (*rule allI*)
unfolding *identity- Π_2 -def* apply (*rule Eq₂I*)

using *D4-2 d2-inject* by *simp*

lemma *lambda-predicates-3-3*[*axiom*]:
 $[[(\lambda^3 (\lambda x y z . \langle F, x^P, y^P, z^P \rangle)) = F]]$
unfolding *axiom-def*
apply (*rule allI*)
unfolding *identity- Π_3 -def* **apply** (*rule Eq3I*)
using *D4-3 d3-inject* by *simp*

lemma *lambda-predicates-4-0*[*axiom*]:
assumes $\bigwedge x. [\langle \mathcal{A}(\varphi x \equiv \psi x) \rangle \text{ in } v]$
shows $[[\langle \lambda^0 (\chi (\iota x. \varphi x)) = \lambda^0 (\chi (\iota x. \psi x)) \rangle]]$
unfolding *axiom-def identity-o-def* **apply** – **apply** (*rule allI*; *rule Eq0I*)
using *TheEqI*[*OF* *assms*[*THEN ActualE*, *THEN EquivE*]] by *auto*

lemma *lambda-predicates-4-1*[*axiom*]:
assumes $\bigwedge x. [\langle \mathcal{A}(\varphi x \equiv \psi x) \rangle \text{ in } v]$
shows $[[\langle (\lambda x . \chi (\iota x. \varphi x) x) = (\lambda x . \chi (\iota x. \psi x) x) \rangle]]$
unfolding *axiom-def identity- Π_1 -def* **apply** – **apply** (*rule allI*; *rule Eq1I*)
using *TheEqI*[*OF* *assms*[*THEN ActualE*, *THEN EquivE*]] by *auto*

lemma *lambda-predicates-4-2*[*axiom*]:
assumes $\bigwedge x. [\langle \mathcal{A}(\varphi x \equiv \psi x) \rangle \text{ in } v]$
shows $[[\langle (\lambda^2 (\lambda x y . \chi (\iota x. \varphi x) x y)) = (\lambda^2 (\lambda x y . \chi (\iota x. \psi x) x y)) \rangle]]$
unfolding *axiom-def identity- Π_2 -def* **apply** – **apply** (*rule allI*; *rule Eq2I*)
using *TheEqI*[*OF* *assms*[*THEN ActualE*, *THEN EquivE*]] by *auto*

lemma *lambda-predicates-4-3*[*axiom*]:
assumes $\bigwedge x. [\langle \mathcal{A}(\varphi x \equiv \psi x) \rangle \text{ in } v]$
shows $[[\langle (\lambda^3 (\lambda x y z . \chi (\iota x. \varphi x) x y z)) = (\lambda^3 (\lambda x y z . \chi (\iota x. \psi x) x y z)) \rangle]]$
unfolding *axiom-def identity- Π_3 -def* **apply** – **apply** (*rule allI*; *rule Eq3I*)
using *TheEqI*[*OF* *assms*[*THEN ActualE*, *THEN EquivE*]] by *auto*

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*]:
 $[[\langle \langle O!, x \rangle \rightarrow \neg(\exists F . \langle x, F \rangle) \rangle]]$
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*]:
 $[[\langle \exists x. \langle A!, x^P \rangle \ \& \ (\forall F . \langle x^P, F \rangle \equiv \varphi F) \rangle]]$
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 κ -proper*)
moreover **have** $[(\forall F . \langle ?x^P, F \rangle \equiv \varphi F) \text{ in } v]$
apply *meta-solver* **unfolding** *en-def*
using *d1.rep-eq d κ -def d κ -proper eval Π_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

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 $\lambda F . \lambda x . \neg (F, x^P)$
 definition *propnot*₂ **where**
 *propnot*₂ \equiv $\lambda F . \lambda^2 (\lambda x y . \neg (F, x^P, y^P))$
 definition *propnot*₃ **where**
 *propnot*₃ \equiv $\lambda F . \lambda^3 (\lambda x y z . \neg (F, x^P, y^P, z^P))$
end

named-theorems *propnot-defs*
declare *propnot*₀-def[*propnot-defs*] *propnot*₁-def[*propnot-defs*]
 *propnot*₂-def[*propnot-defs*] *propnot*₃-def[*propnot-defs*]

8.2 Noncontingent and Contingent Relations

consts *Necessary* :: 'a \Rightarrow o
overloading *Necessary*₀ \equiv *Necessary* :: $\Pi_0 \Rightarrow o$
 *Necessary*₁ \equiv *Necessary* :: $\Pi_1 \Rightarrow o$
 *Necessary*₂ \equiv *Necessary* :: $\Pi_2 \Rightarrow o$
 *Necessary*₃ \equiv *Necessary* :: $\Pi_3 \Rightarrow o$
begin
 definition *Necessary*₀ **where**
 *Necessary*₀ \equiv $\lambda p . \Box p$
 definition *Necessary*₁ :: $\Pi_1 \Rightarrow o$ **where**
 *Necessary*₁ \equiv $\lambda F . \Box (\forall x . (F, x^P))$
 definition *Necessary*₂ **where**
 *Necessary*₂ \equiv $\lambda F . \Box (\forall x y . (F, x^P, y^P))$
 definition *Necessary*₃ **where**
 *Necessary*₃ \equiv $\lambda F . \Box (\forall x y z . (F, x^P, y^P, z^P))$
end

named-theorems *Necessary-defs*
declare *Necessary*₀-def[*Necessary-defs*] *Necessary*₁-def[*Necessary-defs*]
 *Necessary*₂-def[*Necessary-defs*] *Necessary*₃-def[*Necessary-defs*]

consts *Impossible* :: 'a \Rightarrow o
overloading *Impossible*₀ \equiv *Impossible* :: $\Pi_0 \Rightarrow o$
 *Impossible*₁ \equiv *Impossible* :: $\Pi_1 \Rightarrow o$
 *Impossible*₂ \equiv *Impossible* :: $\Pi_2 \Rightarrow o$
 *Impossible*₃ \equiv *Impossible* :: $\Pi_3 \Rightarrow o$
begin
 definition *Impossible*₀ **where**
 *Impossible*₀ \equiv $\lambda p . \Box \neg p$
 definition *Impossible*₁ **where**
 *Impossible*₁ \equiv $\lambda F . \Box (\forall x . \neg (F, x^P))$
 definition *Impossible*₂ **where**
 *Impossible*₂ \equiv $\lambda F . \Box (\forall x y . \neg (F, x^P, y^P))$
 definition *Impossible*₃ **where**
 *Impossible*₃ \equiv $\lambda F . \Box (\forall x y z . \neg (F, x^P, y^P, z^P))$
end

named-theorems *Impossible-defs*

declare *Impossible*_{0-def}[*Impossible-defs*] *Impossible*_{1-def}[*Impossible-defs*]
*Impossible*_{2-def}[*Impossible-defs*] *Impossible*_{3-def}[*Impossible-defs*]

definition *NonContingent* **where**
NonContingent $\equiv \lambda F . (Necessary\ F) \vee (Impossible\ F)$

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

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

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

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

8.3 Null and Universal Objects

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

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

definition *NullObject* :: $\kappa\ (a_0)$ **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* :: $\Pi_1 \Rightarrow \circ$ **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 :: $\kappa \Rightarrow \kappa \Rightarrow \circ$ (**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]:
   $\llbracket [\varphi \text{ in } v]; [\varphi \rightarrow \psi \text{ in } v] \rrbracket \Longrightarrow [\psi \text{ in } v]$ 
  by (simp add: Semantics.T5)

```

9.3 Axioms

```

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

```

9.4 (Modally Strict) Proofs and Derivations

```

lemma vdash-properties-6[no-atp]:
   $\llbracket [\varphi \text{ in } v]; [\varphi \rightarrow \psi \text{ in } v] \rrbracket \Longrightarrow [\psi \text{ in } v]$ 
  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  $\square$ 
    (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 gml-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]:

```

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

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

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

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

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

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

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

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

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

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

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

$$\llbracket [\neg\varphi \text{ in } v]; [\neg\psi \rightarrow \varphi \text{ in } v] \rrbracket \Longrightarrow ([\neg\varphi \text{ in } v] \Longrightarrow [\psi \text{ in } v])$$
using *raa-cor-2* *vdash-properties-10* **by** *blast*

Remark 15. *In contrast to PLM the classical introduction and elimination rules are proven before the tautologies. The statements proven so far are sufficient for the proofs and using the derived rules the tautologies can be derived automatically.*

lemma *intro-elim-1*[PLM]:

$$\llbracket [\varphi \text{ in } v]; [\psi \text{ in } v] \rrbracket \Longrightarrow [\varphi \ \& \ \psi \text{ in } v]$$
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]:

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

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

$$[\varphi \ \& \ \psi \text{ in } v] \Longrightarrow [\psi \text{ in } v]$$
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]:

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

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

$$[\psi \text{ in } v] \Longrightarrow [\varphi \vee \psi \text{ in } v]$$
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]:

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

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

$$\llbracket [\varphi \vee \psi \text{ in } v]; [\neg\varphi \text{ in } v] \rrbracket \Longrightarrow [\psi \text{ in } v]$$
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 \equiv 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 \equiv I)
lemmas $\equiv E$ = intro-elim-6-a intro-elim-6-b intro-elim-6-c
intro-elim-6-d intro-elim-6-e intro-elim-6-f
lemma intro-elim-7[PLM]:

$$[\varphi \text{ in } v] \Longrightarrow [\neg\neg\varphi \text{ in } v]$$
using if-p-then-p modus-tollens-2 by blast
lemmas $\neg\neg I$ = intro-elim-7
lemma intro-elim-8[PLM]:

$$[\neg\neg\varphi \text{ in } v] \Longrightarrow [\varphi \text{ in } v]$$
using if-p-then-p raa-cor-2 by blast
lemmas $\neg\neg E$ = intro-elim-8

context

begin

private lemma NotNotI[PLM-intro]:

$$[\varphi \text{ in } v] \Longrightarrow [\neg(\neg\varphi) \text{ in } v]$$

by (simp add: $\neg\neg I$)

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]:

$$([\varphi \text{ in } v] \Longrightarrow (\bigwedge\psi. [\psi \text{ in } v])) \Longrightarrow [\neg\varphi \text{ in } v]$$

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]:

$$\llbracket [\varphi \text{ in } v]; [\psi \text{ in } v] \rrbracket \Longrightarrow [\varphi \& \psi \text{ in } v]$$

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]:
   $\llbracket [\varphi \text{ in } v] \Longrightarrow [\psi \text{ in } v]; [\psi \text{ in } v] \Longrightarrow [\varphi \text{ in } v] \rrbracket \Longrightarrow [\varphi \equiv \psi \text{ in } v]$ 
  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]:
   $\neg[\varphi \equiv \psi \text{ in } v] \Longrightarrow [\varphi \text{ in } v] \neq [\psi \text{ in } v]$ 
  by (meson NotI contraposition-1  $\equiv I$  vdash-properties-9)

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

private lemma AllI[PLM-intro]:
   $(\bigwedge x . [\varphi x \text{ in } v]) \Longrightarrow [\forall x . \varphi x \text{ in } v]$ 
  using rule-gen by blast
lemma NotAllD[PLM-dest]:
   $\neg[\forall x . \varphi x \text{ in } v] \Longrightarrow (\exists x . \neg[\varphi x \text{ in } v])$ 
  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
Scan.succeed (*Thm.rule-attribute* []
 $(fn \ - \Rightarrow fn \ thm \Rightarrow thm \ RS \ @\{thm \equiv E(1)\})$)
 \gg

attribute-setup *equiv-rl* = \ll
Scan.succeed (*Thm.rule-attribute* []
 $(fn \ - \Rightarrow fn \ thm \Rightarrow thm \ RS \ @\{thm \equiv E(2)\})$)
 \gg

attribute-setup *equiv-sym* = \ll
Scan.succeed (*Thm.rule-attribute* []
 $(fn \ - \Rightarrow fn \ thm \Rightarrow thm \ RS \ @\{thm \ oth-class-taut-3-g[equiv-lr]\})$)
 \gg

attribute-setup *conj1* = \ll
Scan.succeed (*Thm.rule-attribute* []
 $(fn \ - \Rightarrow fn \ thm \Rightarrow thm \ RS \ @\{thm \ \& \ E(1)\})$)

»

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

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

9.7 Identity

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

next
assume 1:  $[(\langle O!, x \rangle \ \& \ \langle O!, y \rangle \ \& \ \Box(\forall F . \langle F, x \rangle \equiv \langle F, y \rangle)) \text{ in } v]$ 
have  $[\forall x y . ((x^P =_E (y^P)) \equiv (\langle O!, x^P \rangle \ \& \ \langle O!, y^P \rangle$ 
   $\ \& \ \Box(\forall F . \langle F, x^P \rangle \equiv \langle F, y^P \rangle)) \text{ in } v]$ 
  unfolding identityE-def identityE-infix-def
  apply (rule lambda-predicates-2-2[axiom-universal, axiom-universal, axiom-instance])
  by show-proper
moreover have  $[\exists \alpha . (\alpha^P) = x \text{ in } v]$ 
  apply (rule cqt-5-mod[where  $\psi = \lambda x . \langle O!, x \rangle$ , axiom-instance, deduction])
  apply (rule SimpleExOrEnc.intros)
  using 1[conj1, conj1] by auto
moreover have  $[\exists \beta . (\beta^P) = y \text{ in } v]$ 
  apply (rule cqt-5-mod[where  $\psi = \lambda y . \langle O!, y \rangle$ , axiom-instance, deduction])
  apply (rule SimpleExOrEnc.intros)
  using 1[conj1, conj2] by auto
ultimately have  $[(x =_E y) \equiv (\langle O!, x \rangle \ \& \ \langle O!, y \rangle$ 
   $\ \& \ \Box(\forall F . \langle F, x \rangle \equiv \langle F, y \rangle)) \text{ in } v]$ 
  using cqt-1-κ[axiom-instance, deduction, deduction] by meson
thus  $[(x =_E y) \text{ in } v]$  using 1  $\equiv E(2)$  by blast

qed
lemma eq-E-simple-2[PLM]:
   $[(x =_E y) \rightarrow (x = y) \text{ in } v]$ 
  unfolding identity-defs by PLM-solver
lemma eq-E-simple-3[PLM]:
   $[(x = y) \equiv (((\langle O!, x \rangle \ \& \ \langle O!, y \rangle \ \& \ \Box(\forall F . \langle F, x \rangle \equiv \langle F, y \rangle))$ 
   $\ \vee \ (\langle A!, x \rangle \ \& \ \langle A!, y \rangle \ \& \ \Box(\forall F . \langle x, F \rangle \equiv \langle y, F \rangle))) \text{ in } v]$ 
  using eq-E-simple-1
  apply – unfolding identity-defs
  by PLM-solver

lemma id-eq-obj-1[PLM]:  $[(x^P) = (x^P) \text{ in } v]$ 
proof –
  have  $[(\Diamond \langle E!, x^P \rangle) \vee (\neg \Diamond \langle E!, x^P \rangle) \text{ in } v]$ 
  using PLM.oth-class-taut-2 by simp
  hence  $[(\Diamond \langle E!, x^P \rangle) \text{ in } v] \vee [(\neg \Diamond \langle E!, x^P \rangle) \text{ in } v]$ 
  using CP ∨ E(1) by blast
moreover {
  assume  $[(\Diamond \langle E!, x^P \rangle) \text{ in } v]$ 
  hence  $[(\lambda x . \Diamond \langle E!, x^P \rangle, x^P) \text{ in } v]$ 
  apply (rule lambda-predicates-2-1[axiom-instance, equiv-rl, rotated])
  by show-proper
  hence  $[(\lambda x . \Diamond \langle E!, x^P \rangle, x^P) \ \& \ (\lambda x . \Diamond \langle E!, x^P \rangle, x^P)]$ 

```

```

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

```

Remark 16. 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  $\nu :: id-eq$ 
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  $o :: id-eq$ 
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 .
  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  $\Pi_2 :: id\text{-}eq$ 
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  $\Pi_3 :: id\text{-}eq$ 
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]:
     $[(x :: 'a :: id\text{-}eq) = x \text{ in } v]$ 
    using id-eq-1 .
  lemma id-eq-2[PLM]:
     $[(x :: 'a :: id\text{-}eq) = y] \rightarrow (y = x) \text{ in } v]$ 

```

```

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

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

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

lemma t-id-t-proper-1[PLM]:
  [τ = τ' → (∃ β . (βP) = τ) in v]
  proof (rule CP)
    assume [τ = τ' in v]
    moreover {
      assume [τ =E τ' in v]
      hence [∃ β . (βP) = τ in v]
      apply -
      apply (rule cqt-5-mod[where ψ=λ τ . τ =E τ', axiom-instance, deduction])
      subgoal unfolding identity-defs by (rule SimpleExOrEnc.intros)
      by simp
    }
    moreover {
      assume [(A!,τ) & (A!,τ') & □(∀ F. {τ,F} ≡ {τ',F}) in v]
      hence [∃ β . (βP) = τ in v]
      apply -
      apply (rule cqt-5-mod[where ψ=λ τ . (A!,τ), axiom-instance, deduction])
      subgoal unfolding identity-defs by (rule SimpleExOrEnc.intros)
      by PLM-solver
    }
    ultimately show [∃ β . (βP) = τ in v] unfolding identityκ-def
      using intro-elim-4-b reductio-aa-1 by blast
  qed

lemma t-id-t-proper-2[PLM]: [τ = τ' → (∃ β . (βP) = τ') in v]
  proof (rule CP)
    assume [τ = τ' in v]
    moreover {
      assume [τ =E τ' in v]
      hence [∃ β . (βP) = τ' in v]
      apply -
      apply (rule cqt-5-mod[where ψ=λ τ' . τ =E τ', axiom-instance, deduction])
      subgoal unfolding identity-defs by (rule SimpleExOrEnc.intros)
      by simp
    }
    moreover {
      assume [(A!,τ) & (A!,τ') & □(∀ F. {τ,F} ≡ {τ',F}) in v]
      hence [∃ β . (βP) = τ' in v]
      apply -
      apply (rule cqt-5-mod[where ψ=λ τ . (A!,τ), axiom-instance, deduction])
      subgoal unfolding identity-defs by (rule SimpleExOrEnc.intros)
      by PLM-solver
    }
    ultimately show [∃ β . (βP) = τ' in v] unfolding identityκ-def
      using intro-elim-4-b reductio-aa-1 by blast

```

qed

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

lemma *id-nec-desc*[*PLM*]:
 $[(\lambda x. \varphi x) = (\lambda x. \psi x)) \equiv \Box((\lambda x. \varphi x) = (\lambda x. \psi x)) \text{ in } v]$
proof (*cases* $[(\exists \alpha. (\alpha^P) = (\lambda x. \varphi x)) \text{ in } v] \wedge [(\exists \beta. (\beta^P) = (\lambda x. \psi x)) \text{ in } v]$)
assume $[(\exists \alpha. (\alpha^P) = (\lambda x. \varphi x)) \text{ in } v] \wedge [(\exists \beta. (\beta^P) = (\lambda x. \psi x)) \text{ in } v]$
then obtain α **and** β **where**
 $[(\alpha^P) = (\lambda x. \varphi x) \text{ in } v] \wedge [(\beta^P) = (\lambda x. \psi x) \text{ in } v]$
apply – **unfolding** *conn-defs* **by** *PLM-solver*
moreover {
moreover have $[(\alpha) = (\beta) \equiv \Box((\alpha) = (\beta)) \text{ in } v]$ **by** *PLM-solver*
ultimately have $[(\lambda x. \varphi x) = (\beta^P) \equiv \Box((\lambda x. \varphi x) = (\beta^P)) \text{ in } v]$
using *l-identity*[**where** $\varphi = \lambda \alpha . (\alpha) = (\beta^P) \equiv \Box((\alpha) = (\beta^P))$, *axiom-instance*]
modus-ponens **unfolding** *identity- ν -def* **by** *metis*
}
ultimately show *?thesis*
using *l-identity*[**where** $\varphi = \lambda \alpha . (\lambda x. \varphi x) = (\alpha)$
 $\equiv \Box((\lambda x. \varphi x) = (\alpha))$, *axiom-instance*]
modus-ponens **by** *metis*
next
assume $\neg[(\exists \alpha. (\alpha^P) = (\lambda x. \varphi x)) \text{ in } v] \wedge [(\exists \beta. (\beta^P) = (\lambda x. \psi x)) \text{ in } v]$
hence $\neg[(\lambda A!. (\lambda x. \varphi x)) \text{ in } v] \wedge \neg[(\lambda x. \varphi x) =_E (\lambda x. \psi x) \text{ in } v]$
 $\vee \neg[(\lambda A!. (\lambda x. \psi x)) \text{ in } v] \wedge \neg[(\lambda x. \varphi x) =_E (\lambda x. \psi x) \text{ in } v]$
unfolding *identity- E -infix-def*
using *cqt-5*[*axiom-instance*] *PLM.contraposition-1 SimpleExOrEnc.intros*
vdash-properties-10 **by** *meson*
hence $\neg[(\lambda x. \varphi x) = (\lambda x. \psi x) \text{ in } v]$
apply – **unfolding** *identity-defs* **by** *PLM-solver*
thus *?thesis* **apply** – **apply** *PLM-solver*
using *qml-2*[*axiom-instance*, *deduction*] **by** *auto*
 qed

9.8 Quantification

lemma *rule-ui*[*PLM*, *PLM-elim*, *PLM-dest*]:
 $[\forall \alpha . \varphi \alpha \text{ in } v] \implies [\varphi \beta \text{ in } v]$
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]:

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

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  $\alpha$  where 1:
       $[\varphi \alpha \ \& \ (\forall \beta. \varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$ 
      by (rule  $\exists E$ )
    {
      fix  $\beta$ 
      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]:
  [ $\mathcal{A}(\varphi \rightarrow \mathcal{A}\varphi)$  in  $v$ ]
  using logic-actual-nec-2[axiom-instance] qml-act-1[axiom-instance]
    ded-thm-cor-3  $\equiv E(2)$  nec-imp-act
  by blast
lemma act-conj-act-3[PLM]:
  [ $(\mathcal{A}\varphi \ \& \ \mathcal{A}\psi) \rightarrow \mathcal{A}(\varphi \ \& \ \psi)$  in  $v$ ]
  unfolding conn-defs
  by (metis logic-actual-nec-2[axiom-instance]
    logic-actual-nec-1[axiom-instance]
     $\equiv E(2)$  CP  $\equiv E(4)$  reductio-aa-2
    vdash-properties-10)
lemma act-conj-act-4[PLM]:
  [ $\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)$  in  $v$ ]
  unfolding equiv-def
  by (PLM-solver PLM-intro: act-conj-act-3[where  $\varphi = \mathcal{A}\varphi \rightarrow \varphi$ 
    and  $\psi = \varphi \rightarrow \mathcal{A}\varphi$ , deduction])
lemma closure-act-1a[PLM]:
  [ $\mathcal{A}\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)$  in  $v$ ]
  using logic-actual-nec-4[axiom-instance]
    act-conj-act-4  $\equiv E(1)$ 
  by blast
lemma closure-act-1b[PLM]:
  [ $\mathcal{A}\mathcal{A}\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)$  in  $v$ ]
  using logic-actual-nec-4[axiom-instance]
    act-conj-act-4  $\equiv E(1)$ 
  by blast
lemma closure-act-1c[PLM]:
  [ $\mathcal{A}\mathcal{A}\mathcal{A}\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)$  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[necessitation-averse-axiom-instance] 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]:
  [ $\mathcal{A}(\forall x . \varphi x)$  in  $v$ ]  $\implies$  [ $\forall x . \mathcal{A}(\varphi x)$  in  $v$ ]
  using logic-actual-nec-3[axiom-instance, equiv-lr] .
lemma NotActualForallD[PLM-dest]:
   $\neg[\mathcal{A}(\forall x . \varphi x)$  in  $v$ ]  $\implies$   $\neg[\forall x . \mathcal{A}(\varphi x)$  in  $v$ ]
  using ActualForallI by blast

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

lemma ANeg-1[PLM]:
  [ $\neg\mathcal{A}\varphi \equiv \neg\varphi$  in  $dw$ ]
  by PLM-solver
lemma ANeg-2[PLM]:
  [ $\neg\mathcal{A}\neg\varphi \equiv \varphi$  in  $dw$ ]
  by PLM-solver
lemma Act-Basic-1[PLM]:
  [ $\mathcal{A}\varphi \vee \mathcal{A}\neg\varphi$  in  $v$ ]
  by PLM-solver
lemma Act-Basic-2[PLM]:
  [ $\mathcal{A}(\varphi \ \& \ \psi) \equiv (\mathcal{A}\varphi \ \& \ \mathcal{A}\psi)$  in  $v$ ]
  by PLM-solver
lemma Act-Basic-3[PLM]:
  [ $\mathcal{A}(\varphi \equiv \psi) \equiv ((\mathcal{A}(\varphi \rightarrow \psi)) \ \& \ (\mathcal{A}(\psi \rightarrow \varphi)))$  in  $v$ ]
  by PLM-solver
lemma Act-Basic-4[PLM]:
  [ $(\mathcal{A}(\varphi \rightarrow \psi) \ \& \ \mathcal{A}(\psi \rightarrow \varphi)) \equiv (\mathcal{A}\varphi \equiv \mathcal{A}\psi)$  in  $v$ ]
  by PLM-solver
lemma Act-Basic-5[PLM]:
  [ $\mathcal{A}(\varphi \equiv \psi) \equiv (\mathcal{A}\varphi \equiv \mathcal{A}\psi)$  in  $v$ ]
  by PLM-solver
lemma Act-Basic-6[PLM]:
  [ $\Diamond\varphi \equiv \mathcal{A}(\Diamond\varphi)$  in  $v$ ]
  unfolding diamond-def by PLM-solver
lemma Act-Basic-7[PLM]:
  [ $\mathcal{A}\varphi \equiv \Box\mathcal{A}\varphi$  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 $[\mathcal{A}\neg(\forall \alpha . \neg\varphi \alpha) \equiv \neg\mathcal{A}(\forall \alpha . \neg\varphi \alpha) \text{ in } v]$
 using *logic-actual-nec-1*[*axiom-instance*] by *blast*
 ultimately have $[\mathcal{A}\neg(\forall \alpha . \neg\varphi \alpha) \equiv \neg(\forall \alpha . \mathcal{A}\neg\varphi \alpha) \text{ in } v]$
 using $\equiv E(5)$ by *auto*
 moreover {
 have $[\forall \alpha . \mathcal{A}\neg\varphi \alpha \equiv \neg\mathcal{A}\varphi \alpha \text{ in } v]$
 using *logic-actual-nec-1*[*axiom-universal*, *axiom-instance*] by *blast*
 hence $[(\forall \alpha . \mathcal{A}\neg\varphi \alpha) \equiv (\forall \alpha . \neg\mathcal{A}\varphi \alpha) \text{ in } v]$
 using *cqt-basic-3*[*deduction*] by *fast*
 hence $[(\neg(\forall \alpha . \mathcal{A}\neg\varphi \alpha)) \equiv \neg(\forall \alpha . \neg\mathcal{A}\varphi \alpha) \text{ in } v]$
 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*]:
 $[(\forall z . \mathcal{A}\varphi z \equiv z = x) \equiv (\forall z . \varphi z \equiv z = x) \text{ in } dw]$
 by *PLM-solver*

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

lemma *hintikka*[*PLM*]:
 $[(x^P = (\iota x . \varphi x)) \equiv (\varphi x \ \& \ (\forall z . \varphi z \rightarrow z = x)) \text{ in } dw]$
 proof –
 have $[(\forall z . \varphi z \equiv z = x) \equiv (\varphi x \ \& \ (\forall z . \varphi z \rightarrow z = x)) \text{ in } dw]$
 unfolding *identity-ν-def* apply *PLM-solver* using *id-eq-obj-1* apply *simp*
 using *l-identity*[*where* $\varphi = \lambda x . \varphi x$, *axiom-instance*,
deduction, *deduction*]
 using *id-eq-obj-2*[*deduction*] unfolding *identity-ν-def* by *fastforce*
 thus *?thesis* using $\equiv E(5)$ *fund-cont-desc* by *blast*
 qed

lemma *russell-axiom-a*[*PLM*]:
 $[(\langle F, \iota x . \varphi x \rangle) \equiv (\exists x . \varphi x \ \& \ (\forall z . \varphi z \rightarrow z = x) \ \& \ \langle F, x^P \rangle) \text{ in } dw]$
 (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 $\varphi = \lambda x . \langle F, x \rangle$,
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  $[\alpha^P = (\iota x . \varphi x) \text{ in } dw] \wedge [\llbracket F, \alpha^P \rrbracket \text{ in } dw]$ 
    using hintikka[equiv-rl] &E by blast
  hence  $[?lhs \text{ in } dw]$ 
    using l-identity[axiom-instance, deduction, deduction]
    by blast
}
ultimately show ?thesis by PLM-solver
qed

lemma russell-axiom-g[PLM]:
   $[\llbracket \iota x . \varphi x, F \rrbracket \equiv (\exists x . \varphi x \ \& \ (\forall z . \varphi z \rightarrow z = x) \ \& \ \llbracket \alpha^P, F \rrbracket) \text{ in } dw]$ 
  (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] SimpleExOrEnc.intros by blast
    then obtain  $\alpha$  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  $\varphi = \lambda x . x = \alpha^P$ ,
        axiom-instance, deduction, deduction]
        id-eq-obj-1[where  $x = \alpha$ ] 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  $\varphi = \lambda x . \llbracket x, F \rrbracket$ ,
      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  $\exists I$ [where  $\alpha = \alpha$ ] by (simp add: identity-defs)
  }
  moreover {
    assume  $[?rhs \text{ in } dw]$ 
    then obtain  $\alpha$  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  $\psi$ 
  shows  $[\psi (\iota x . \varphi x) \equiv (\exists x . \varphi x \ \& \ (\forall z . \varphi z \rightarrow z = x) \ \& \ \psi (\alpha^P)) \text{ in } dw]$ 
  (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  $\alpha$  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  $\varphi = \lambda x . x = \alpha^P$ ,
        axiom-instance, deduction, deduction]
        id-eq-obj-1[where  $x = \alpha$ ] by auto
    hence  $[\psi (\alpha^P) \text{ in } dw]$ 

```

```

    using 1 l-identity[where  $\beta = \alpha^P$  and  $\alpha = \iota x. \varphi x$  and  $\varphi = \lambda x. \psi x$ ,
      axiom-instance, deduction, deduction] by auto
  with  $\beta$  have  $[\varphi \alpha \ \& \ (\forall z. \varphi z \rightarrow z = \alpha) \ \& \ \psi(\alpha^P) \text{ in } dw]$ 
    using &I by auto
  hence  $[?rhs \text{ in } dw]$  using  $\exists I$ [where  $\alpha = \alpha$ ] by (simp add: identity-defs)
}
moreover {
  assume  $[?rhs \text{ in } dw]$ 
  then obtain  $\alpha$  where  $\ell_4$ :
     $[\varphi \alpha \ \& \ (\forall z. \varphi z \rightarrow z = \alpha) \ \& \ \psi(\alpha^P) \text{ in } dw]$ 
    using  $\exists E$  by auto
  hence  $[\alpha^P = (\iota x. \varphi x) \text{ in } dw] \wedge [\psi(\alpha^P) \text{ in } dw]$ 
    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 unique-exists[PLM]:
 $[(\exists y. y^P = (\iota x. \varphi x)) \equiv (\exists !x. \varphi x) \text{ in } dw]$
proof((rule $\equiv I$, rule CP, rule-tac[2] CP))
 assume $[\exists y. y^P = (\iota x. \varphi x) \text{ in } dw]$
 then obtain α where
 $[\alpha^P = (\iota x. \varphi x) \text{ in } dw]$
 by (rule $\exists E$)
 hence $[\varphi \alpha \ \& \ (\forall \beta. \varphi \beta \rightarrow \beta = \alpha) \text{ in } dw]$
 using hintikka[equiv-lr] by auto
 thus $[\exists !x. \varphi x \text{ in } dw]$
 unfolding exists-unique-def using $\exists I$ by fast
next
 assume $[\exists !x. \varphi x \text{ in } dw]$
 then obtain α where
 $[\varphi \alpha \ \& \ (\forall \beta. \varphi \beta \rightarrow \beta = \alpha) \text{ in } dw]$
 unfolding exists-unique-def by (rule $\exists E$)
 hence $[\alpha^P = (\iota x. \varphi x) \text{ in } dw]$
 using hintikka[equiv-rl] by auto
 thus $[\exists y. y^P = (\iota x. \varphi x) \text{ in } dw]$
 using $\exists I$ by fast
qed

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

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

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

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

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

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

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

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

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

apply blast
 using l-identity[where $\alpha=x$, axiom-instance, deduction, deduction]
 id-eq-2[where 'a= ν , 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 – 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 – 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: $[\mathcal{A}\psi x \ \& \ (\forall z. \mathcal{A}\psi z \rightarrow z = x) \text{ in } v]$
 using nec-hintikka-scheme[equiv-lr] by auto
 hence 2: $[\mathcal{A}\psi x \text{ in } v] \wedge [(\forall z. \mathcal{A}\psi z \rightarrow z = x) \text{ in } v]$
 using $\&E$ by blast
 {
 fix z
 {
 assume $[\mathcal{A}\varphi z \text{ in } v]$
 hence $[\mathcal{A}\psi z \text{ in } v]$
 using assms[where $x=z$]
 apply – by PLM-solver
 moreover have $[\mathcal{A}\psi z \rightarrow z = x \text{ in } v]$
 using 2 cqt-1[axiom-instance,deduction] by auto
 ultimately have $[z = x \text{ in } v]$
 using vdash-properties-10 by auto
 }
 hence $[\mathcal{A}\varphi z \rightarrow z = x \text{ in } v]$ by (rule CP)
 }
 hence $[(\forall z. \mathcal{A}\varphi z \rightarrow z = x) \text{ in } v]$ by (rule $\forall I$)
 moreover have $[\mathcal{A}\varphi x \text{ in } v]$
 using 1[conj1] assms[where $x=x$]
 apply – by PLM-solver
 ultimately have $[\mathcal{A}\varphi x \ \& \ (\forall z. \mathcal{A}\varphi z \rightarrow z = x) \text{ in } v]$
 by PLM-solver


```

    hence  $[x^P = (\iota x. \varphi x) \text{ in } v]$ 
      using nec-hintikka-scheme [where  $\varphi = \varphi, \text{equiv-rl}$ ]
      by auto
  }
  ultimately show  $[x^P = (\iota x. \varphi x) \equiv (x^P) = (\iota x. \psi x) \text{ in } v]$ 
    using  $\equiv I$  CP by auto
qed

```

lemma *UniqueAux*:

```

assumes  $[(\mathcal{A}\varphi (\alpha::\nu) \ \& \ (\forall z. \mathcal{A}(\varphi z) \rightarrow z = \alpha)) \text{ in } v]$ 
shows  $[(\forall z. (\mathcal{A}(\varphi z) \equiv (z = \alpha))) \text{ in } v]$ 
proof -
{
  fix  $z$ 
  {
    assume  $[\mathcal{A}(\varphi z) \text{ in } v]$ 
    hence  $[z = \alpha \text{ in } v]$ 
      using assms [conj2, THEN cqt-1 [where  $\alpha = z$ ,
        axiom-instance, deduction],
        deduction] by auto
  }
  moreover {
    assume  $[z = \alpha \text{ in } v]$ 
    hence  $[\alpha = z \text{ in } v]$ 
      unfolding identity- $\nu$ -def
      using id-eq-obj-2 [deduction] by fast
    hence  $[\mathcal{A}(\varphi z) \text{ in } v]$  using assms [conj1]
      using l-identity [axiom-instance, deduction,
        deduction] by fast
  }
  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  $[(\psi (\iota x. \varphi x)) \equiv (\exists x. (\mathcal{A}\varphi x \ \& \ (\forall z. \mathcal{A}(\varphi z) \rightarrow z = x)) \ \& \ \psi (x^P)) \text{ in } v]$ 
(is  $[?lhs \equiv ?rhs \text{ in } v]$ )
proof -
{
  assume 1:  $[?lhs \text{ in } v]$ 
  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  $\beta$  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 [ $?rhs$  in  $v$ ]
  using  $\exists I$ [where  $\alpha=\alpha$ ]
  by (simp add: identity-defs)
}
moreover {
  assume [ $?rhs$  in  $v$ ]
  then obtain  $\alpha$  where  $\beta$ :
     $[(\mathcal{A}\varphi \alpha \ \& \ (\forall z . \mathcal{A}(\varphi z) \rightarrow z = \alpha)) \ \& \ \psi(\alpha^P) \text{ in } v]$ 
    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]
     $\beta$ [conj2] by blast
  hence [ $?lhs$  in  $v$ ]
    using l-identity[axiom-instance, deduction,
      deduction]
    by fast
}
ultimately show  $?thesis$  by PLM-solver
qed

```

lemma *actual-desc-1*[PLM]:
 $[(\exists y . (y^P) = (\iota x . \varphi x)) \equiv (\exists! x . \mathcal{A}(\varphi x)) \text{ in } v] \text{ (is } [?lhs \equiv ?rhs \text{ in } v])$
proof –
{
 assume [$?lhs$ in v]
 then obtain α where
 $[(\alpha^P) = (\iota x . \varphi x)) \text{ in } v]$
 by (rule $\exists E$)
 hence $[(\lambda! . (\iota x . \varphi x)) \text{ in } v] \vee [(\alpha^P) =_E (\iota x . \varphi x) \text{ in } v]$
 apply – 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 $\psi=\lambda x . (\lambda! . x)$, equiv-lr, THEN $\exists E$]
 using nec-russell-axiom[where $\psi=\lambda x . (\alpha^P) =_E x$, equiv-lr, THEN $\exists E$]
 using SimpleExOrEnc.intros unfolding identity_E-infix-def
 by (meson &E)
 hence [$?rhs$ in v] unfolding exists-unique-def by (rule $\exists I$)
}
moreover {
 assume [$?rhs$ in v]
 then obtain x where
 $[(\mathcal{A}\varphi x \ \& \ (\forall z . \mathcal{A}(\varphi z) \rightarrow z = x)) \text{ in } v]$
 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 α **where** 1:
 $[\Box \alpha \ \& \ (\forall \beta. \Box(\varphi \beta) \rightarrow \beta = \alpha) \text{ in } v]$
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]*:
 $[\Box(\varphi \ \& \ \psi) \rightarrow \Box(\varphi \equiv \psi) \text{ in } v]$
using *KBasic-7 KBasic-3*
by (*metis equiv-def PLM.ded-thm-cor-3 $\& E(1)$*)

lemma *KBasic-9[PLM]*:
 $[\Box((\neg \varphi) \ \& \ (\neg \psi)) \rightarrow \Box(\varphi \equiv \psi) \text{ in } v]$
proof (*rule CP*)
assume $[\Box((\neg \varphi) \ \& \ (\neg \psi)) \text{ in } v]$
hence $[\Box((\neg \varphi) \equiv (\neg \psi)) \text{ in } v]$
using *KBasic-8 vdash-properties-10* **by** *blast*
moreover have $\bigwedge v. [((\neg \varphi) \equiv (\neg \psi)) \rightarrow (\varphi \equiv \psi) \text{ in } v]$

using $CP \equiv E(2)$ *oth-class-taut-5-d* by blast
 ultimately show $[\Box(\varphi \equiv \psi) \text{ in } v]$
 using *RM-1 PLM.vdash-properties-10* by blast
 qed

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

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

$[\Box(\psi \equiv \chi) \text{ in } v] \implies [(\psi \rightarrow \Theta) \equiv (\chi \rightarrow \Theta) \text{ in } v]$
 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]:

$[\Box(\psi \equiv \chi) \text{ in } v] \implies [(\Theta \rightarrow \psi) \equiv (\Theta \rightarrow \chi) \text{ in } v]$
 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]:

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

$[\Box(\psi \equiv \chi) \text{ in } v] \implies [\mathcal{A}\psi \equiv \mathcal{A}\chi \text{ in } v]$
 using *Act-Basic-5 \equiv E(1) nec-imp-act*
vdash-properties-10
 by blast

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

$[\Box(\psi \equiv \chi) \text{ in } v] \implies [\Box\psi \equiv \Box\chi \text{ in } v]$
 using *KBasic-6 \equiv I \equiv E(1) vdash-properties-9*
 by blast

named-theorems *Substable-intros*

definition *Substable* :: $('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow ('a \Rightarrow \text{bool}) \Rightarrow \text{bool}$

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

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

Substable $\text{cond } (\lambda \varphi . \Theta)$
 unfolding *Substable-def* using *oth-class-taut-4-a* by blast

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

assumes *Substable* $\text{cond } \psi$
 shows *Substable* $\text{cond } (\lambda \varphi . \neg(\psi \varphi))$
 using *assms unfolding Substable-def*
 using *rule-sub-lem-1-a RN-2 \equiv E oth-class-taut-5-d* by metis

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

assumes *Substable* $\text{cond } \psi$
 and *Substable* $\text{cond } \chi$
 shows *Substable* $\text{cond } (\lambda \varphi . \psi \varphi \rightarrow \chi \varphi)$
 using *assms unfolding Substable-def*
 by (metis $\equiv I CP$ *intro-elim-6-a intro-elim-6-b*)

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

assumes *Substable* $\text{cond } \psi$
 shows *Substable* $\text{cond } (\lambda \varphi . \Box(\psi \varphi))$
 using *assms unfolding Substable-def*
 using *rule-sub-lem-1-f RN* by meson

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

assumes *Substable* $\text{cond } \psi$
 shows *Substable* $\text{cond } (\lambda \varphi . \mathcal{A}(\psi \varphi))$
 using *assms unfolding Substable-def*
 using *rule-sub-lem-1-e RN* by meson

```

lemma Substable-intro-all[Substable-intros]:
  assumes  $\forall x . \text{Substable cond } (\psi x)$ 
  shows  $\text{Substable cond } (\lambda \varphi . \forall x . \psi x \varphi)$ 
  using assms unfolding Substable-def
  by (simp add: RN rule-sub-lem-1-d)

named-theorems Substable-Cond-defs
end

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

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

instantiation fun :: (type, Substable) Substable
begin
  definition Substable-Cond-fun where [PLM.Substable-Cond-defs]:
     $\text{Substable-Cond-fun} \equiv \lambda \varphi \psi . \forall x . \text{Substable-Cond } (\varphi x) (\psi x)$ 
  instance proof
    interpret PLM .
    fix  $\varphi :: ('a \Rightarrow 'b) \Rightarrow o$  and  $\psi \chi :: 'a \Rightarrow 'b$  and  $\Theta v$ 
    assume Substable Substable-Cond  $\varphi$ 
    moreover assume Substable-Cond  $\psi \chi$ 
    ultimately have  $[\varphi \psi \equiv \varphi \chi \text{ in } v]$ 
    unfolding Substable-def by blast
    hence  $[\varphi \psi \text{ in } v] = [\varphi \chi \text{ in } v]$  using  $\equiv E$  by blast
    moreover assume  $\Theta [\varphi \psi \text{ in } v]$ 
    ultimately show  $\Theta [\varphi \chi \text{ in } v]$  by simp
  qed
end

context PLM
begin

  lemma Substable-intro-equiv[Substable-intros]:
    assumes Substable cond  $\psi$ 
    and Substable cond  $\chi$ 
    shows Substable cond  $(\lambda \varphi . \psi \varphi \equiv \chi \varphi)$ 
    unfolding conn-defs by (simp add: assms Substable-intros)
  lemma Substable-intro-conj[Substable-intros]:
    assumes Substable cond  $\psi$ 
    and Substable cond  $\chi$ 
    shows Substable cond  $(\lambda \varphi . \psi \varphi \ \& \ \chi \varphi)$ 
    unfolding conn-defs by (simp add: assms Substable-intros)
  lemma Substable-intro-disj[Substable-intros]:

```

```

assumes Substable cond  $\psi$ 
and Substable cond  $\chi$ 
shows Substable cond  $(\lambda \varphi . \psi \varphi \vee \chi \varphi)$ 
unfolding conn-defs by (simp add: assms Substable-intros)
lemma Substable-intro-diamond[Substable-intros]:
assumes Substable cond  $\psi$ 
shows Substable cond  $(\lambda \varphi . \Diamond(\psi \varphi))$ 
unfolding conn-defs by (simp add: assms Substable-intros)
lemma Substable-intro-exist[Substable-intros]:
assumes  $\forall x . \text{Substable cond } (\psi x)$ 
shows Substable cond  $(\lambda \varphi . \exists x . \psi x \varphi)$ 
unfolding conn-defs by (simp add: assms Substable-intros)

lemma Substable-intro-id-o[Substable-intros]:
Substable Substable-Cond  $(\lambda \varphi . \varphi)$ 
unfolding Substable-def Substable-Cond-o-def by blast
lemma Substable-intro-id-fun[Substable-intros]:
assumes Substable Substable-Cond  $\psi$ 
shows Substable Substable-Cond  $(\lambda \varphi . \psi (\varphi x))$ 
using assms unfolding Substable-def Substable-Cond-fun-def
by blast

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

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

method PLM-autosubst1 =
  (match premises in  $\bigwedge v x . [\psi x \equiv \chi x \text{ in } v]$ 
    for  $\psi::'a::\text{type} \Rightarrow o$  and  $\chi::'a \Rightarrow o \Rightarrow$ 
     $\langle \text{match conclusion in } \Theta [\varphi \chi \text{ in } v] \text{ for } \Theta \varphi \text{ and } v \Rightarrow$ 
       $\langle (\text{rule rule-sub-nec}[\text{where } \Theta=\Theta \text{ and } \chi=\chi \text{ and } \psi=\psi \text{ and } \varphi=\varphi \text{ and } v=v],$ 
         $((\text{fast intro: Substable-intros, } ((\text{assumption})+)?)+; \text{fail}),$ 
        unfold Substable-Cond-defs) $\rangle \rangle$ )

method PLM-autosubst2 =
  (match premises in  $\bigwedge v x y . [\psi x y \equiv \chi x y \text{ in } v]$ 
    for  $\psi::'a::\text{type} \Rightarrow 'a \Rightarrow o$  and  $\chi::'a::\text{type} \Rightarrow 'a \Rightarrow o \Rightarrow$ 
     $\langle \text{match conclusion in } \Theta [\varphi \chi \text{ in } v] \text{ for } \Theta \varphi \text{ and } v \Rightarrow$ 
       $\langle (\text{rule rule-sub-nec}[\text{where } \Theta=\Theta \text{ and } \chi=\chi \text{ and } \psi=\psi \text{ and } \varphi=\varphi \text{ and } v=v],$ 
         $((\text{fast intro: Substable-intros, } ((\text{assumption})+)?)+; \text{fail}),$ 
        unfold Substable-Cond-defs) $\rangle \rangle$ )

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

lemma rule-sub-nec[PLM]:
assumes Substable Substable-Cond  $\varphi$ 
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 Substable-Cond-defs*
using $\equiv I \text{ CP } \equiv E(1) \equiv E(2)$ **by** *meson*
thus $\Theta [\varphi \psi \text{ in } v] \implies \Theta [\varphi \chi \text{ in } v]$ **by** *auto*
qed

lemma *rule-sub-nec1[PLM]*:
assumes *Substable Substable-Cond* φ
shows $(\bigwedge v x.[(\psi x \equiv \chi x) \text{ in } v]) \implies \Theta [\varphi \psi \text{ in } v] \implies \Theta [\varphi \chi \text{ in } v]$
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 Substable-def Substable-Cond-defs*
using $\equiv I \text{ CP } \equiv E(1) \equiv E(2)$ **by** *metis*
thus $\Theta [\varphi \psi \text{ in } v] \implies \Theta [\varphi \chi \text{ in } v]$ **by** *auto*
qed

lemma *rule-sub-nec2[PLM]*:
assumes *Substable Substable-Cond* φ
shows $(\bigwedge v x y.[\psi x y \equiv \chi x y \text{ in } v]) \implies \Theta [\varphi \psi \text{ in } v] \implies \Theta [\varphi \chi \text{ in } v]$
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 Substable-def Substable-Cond-defs*
using $\equiv I \text{ CP } \equiv E(1) \equiv E(2)$ **by** *metis*
thus $\Theta [\varphi \psi \text{ in } v] \implies \Theta [\varphi \chi \text{ in } v]$ **by** *auto*
qed

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

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

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

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

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

lemma *rule-sub-remark-4*:
 assumes $\bigwedge v x. [\neg(\neg(P, x^P))] \equiv (P, x^P) \text{ in } v]$
 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 *DfBox* **by** (*rule* $\equiv E(6)$)

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

apply – **apply** (*PLM-subst-method* $\Box(\neg\varphi) \ \neg(\Diamond\varphi)$)

apply (*simp add: KBasic2-4*)

apply (*PLM-subst-method* $\Box(\neg\psi) \ \neg(\Diamond\psi)$)

apply (*simp add: KBasic2-4*)

unfolding *diamond-def* **by** *assumption*

hence $[(\neg(\Diamond(\varphi \vee \psi))) \equiv ((\neg(\Diamond\varphi)) \ \& \ (\neg(\Diamond\psi))) \text{ in } v]$

apply – **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 – **apply** (*PLM-subst-method* $\neg(\neg(\Diamond(\varphi \vee \psi))) \ \Diamond(\varphi \vee \psi)$)

using *oth-class-taut-4-b[equiv-sym]* **by** *auto*

thus *?thesis*

apply – **apply** (*PLM-subst-method* $\neg((\neg(\Diamond\varphi)) \ \& \ (\neg(\Diamond\psi))) \ (\Diamond\varphi) \vee (\Diamond\psi)$)

using *oth-class-taut-6-b[equiv-sym]* **by** *auto*

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** *simp*

apply (*PLM-subst-method* $(\neg\varphi) \vee \psi \ \varphi \rightarrow \psi$)

using *oth-class-taut-5-k[equiv-sym]* **apply** *simp*

apply (*PLM-subst-method* $\Diamond(\neg\varphi) \ \neg(\Box\varphi)$)

using *KBasic2-2[equiv-sym]* **apply** *simp*

using *KBasic2-6* .

```

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

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

lemma KBasic2-12[PLM]: [ $\Box(\varphi \vee \psi) \rightarrow (\Box\varphi \vee \Diamond\psi)$  in v]
  proof –
    have [ $\Box(\psi \vee \varphi) \rightarrow (\Box(\neg\psi) \rightarrow \Box\varphi)$  in v]
      using CP RM-1-b  $\vee E(2)$  by blast
    hence [ $\Box(\psi \vee \varphi) \rightarrow (\Diamond\psi \vee \Box\varphi)$  in v]
      unfolding diamond-def disj-def
      by (meson CP  $\neg\neg E$  vdash-properties-6)
    thus ?thesis apply –
      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]:
  [ $\varphi \rightarrow \Diamond\varphi$  in v]
  unfolding diamond-def
  apply (subst contraposition-1)
  apply (PLM-subst-method  $\Box\neg\varphi \neg\neg\Box\neg\varphi$ )
  apply (simp add: PLM.oth-class-taut-4-b)
  using qml-2[where  $\varphi = \neg\varphi$ , axiom-instance]
  by simp
lemmas  $T\Diamond = TBasic$ 

lemma S5Basic-1[PLM]:
  [ $\Diamond\Box\varphi \rightarrow \Box\varphi$  in v]
  proof (rule CP)
    assume [ $\Diamond\Box\varphi$  in v]
    hence [ $\neg\Box\Diamond\neg\varphi$  in v]
      using KBasic2-10[equiv-lr] by simp
    moreover have [ $\Diamond(\neg\varphi) \rightarrow \Box\Diamond(\neg\varphi)$  in v]
      by (simp add: qml-3[axiom-instance])
    ultimately have [ $\neg\Diamond\neg\varphi$  in v]
      by (simp add: PLM.modus-tollens-1)
    thus [ $\Box\varphi$  in v]
      unfolding diamond-def apply –
        apply (PLM-subst-method  $\neg\neg\varphi \varphi$ )
        using oth-class-taut-4-b[equiv-sym] apply simp
        unfolding diamond-def using oth-class-taut-4-b[equiv-rl]
        by simp
  qed
lemmas  $S5\Diamond = S5Basic-1$ 

lemma S5Basic-2[PLM]:
  [ $\Box\varphi \equiv \Diamond\Box\varphi$  in v]
  using  $S5\Diamond$   $T\Diamond \equiv I$  by blast

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

```

lemma *S5Basic-4*[PLM]:
 $[\varphi \rightarrow \Box\Diamond\varphi \text{ in } v]$
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* \Diamond **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** *simp*
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** *simp*
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** *simp*
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** *simp*
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** *simp*
apply (*PLM-subst-method* $\Box\neg\varphi \neg\Diamond\varphi$)
using *KBasic2-4* **apply** *simp*
apply (*PLM-subst-method* $(\neg\Box((\neg\varphi) \vee \Box(\neg\psi))) (\Diamond(\neg((\neg\varphi) \vee (\Box(\neg\psi))))))$

```

    unfolding diamond-def
    apply (simp add: RN oth-class-taut-4-b rule-sub-lem-1-a rule-sub-lem-1-f)
    using 1 by assumption
  show ?thesis
    apply (PLM-subst-method  $\neg((\neg\varphi) \vee (\neg\Diamond\psi)) \varphi \ \& \ \Diamond\psi$ )
    using oth-class-taut-6-a[equiv-sym] apply simp
    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 simp
    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 simp
  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 $\Box(\Diamond\varphi \rightarrow \varphi)$ in v
 using *S5Basic-14*[*equiv-lr*] by auto
 hence $\Diamond\varphi \rightarrow \varphi$ in v
 using *qml-2*[*axiom-instance*, *deduction*] by auto
 moreover from 1 have $\varphi \rightarrow \Box\varphi$ in v
 using *qml-2*[*axiom-instance*, *deduction*] by auto
 ultimately have $\Diamond\varphi \rightarrow \Box\varphi$ in v
 using *ded-thm-cor-3* by auto
 moreover have $\Box\varphi \rightarrow \Diamond\varphi$ in v
 using *qml-2*[*axiom-instance*] *T* \Diamond
 by (rule *ded-thm-cor-3*)
 ultimately show $\Diamond\varphi \equiv \Box\varphi$ in v
 by (rule $\equiv I$)
 qed

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

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

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

```

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

```

lemma *BFs-4* [*PLM*]:

```

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

```

lemma *sign-S5-thm-1* [*PLM*]:

```

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

```

lemma *sign-S5-thm-2* [*PLM*]:

```

[ $\Diamond(\forall \alpha. \varphi \ \alpha) \rightarrow (\forall \alpha. \Diamond(\varphi \ \alpha))$ ] in v]
proof -
  {
    fix  $\alpha$ 
    {
      fix v
      have [( $\forall \alpha. \varphi \ \alpha$ )  $\rightarrow \varphi \ \alpha$ ] in v]
        using cqt-orig-1 by metis
    }
    hence [ $\Diamond(\forall \alpha. \varphi \ \alpha) \rightarrow \Diamond(\varphi \ \alpha)$ ] in v]
      using RM-2 by metis
  }

```



```

}
hence  $[\forall \alpha . \Diamond(\forall \alpha . \varphi \alpha) \rightarrow \Diamond(\varphi \alpha) \text{ in } v]$ 
  using  $\forall I$  by metis
thus ?thesis
  using cqt-orig-2[deduction] by metis
qed
lemmas Buridan $\Diamond = \text{sign-S5-thm-2}$ 

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]:
 $[(\Box(\forall \alpha . \varphi \alpha \equiv \psi \alpha)) \ \& \ (\Box(\forall \alpha . \psi \alpha \equiv \chi \alpha))] \rightarrow (\Box(\forall \alpha . \varphi \alpha \equiv \chi \alpha)) \text{ in } v]$ 
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 by 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=v']
      modus-ponens unfolding identity- $\nu$ -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]:
   $[\Diamond \llbracket x, F \rrbracket \equiv \Box \llbracket x, F \rrbracket \text{ in } v]$ 
  using encoding[axiom-instance] RN
    sc-eq-box-box-1 modus-ponens by blast

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

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

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

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

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

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

lemma en-eq-8[PLM]:
   $[\Diamond(\neg \llbracket x, F \rrbracket) \equiv (\neg \llbracket x, F \rrbracket) \text{ in } v]$ 
  unfolding diamond-def apply (PLM-subst-method  $\llbracket x, F \rrbracket \neg \neg \llbracket x, F \rrbracket$ )
  using oth-class-taut-4-b apply simp
  apply (PLM-subst-method  $\llbracket x, F \rrbracket \Box \llbracket x, F \rrbracket$ )
  using en-eq-2 apply simp
  using oth-class-taut-4-a by assumption

lemma en-eq-9[PLM]:
   $[\Diamond(\neg \llbracket x, F \rrbracket) \equiv \Box(\neg \llbracket x, F \rrbracket) \text{ in } v]$ 
  using en-eq-8 en-eq-7  $\equiv E(5)$  by blast

lemma en-eq-10[PLM]:
   $[\mathcal{A} \llbracket x, F \rrbracket \equiv \llbracket x, F \rrbracket \text{ in } v]$ 
  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 *IsProperInX* φ
and *IsProperInX* ψ
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 *IsProperInXY* φ
and *IsProperInXY* ψ
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 *IsProperInXYZ* φ
and *IsProperInXYZ* ψ
and $\bigwedge x y z. [\varphi(x^P)(y^P)(z^P) \equiv \psi(x^P)(y^P)(z^P) \text{ in } v]$
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 *IsProperInX* φ
and *IsProperInX* ψ
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 *IsProperInXY* φ
and *IsProperInXY* ψ
shows $[(\Box(\forall x y. \varphi(x^P)(y^P) \equiv \psi(x^P)(y^P))) \rightarrow (\Box(\forall x y. (\lambda^2(\lambda x y. \varphi(x^P)(y^P)), x^P, y^P) \equiv (\lambda^2(\lambda x y. \psi(x^P)(y^P)), x^P, y^P))) \text{ in } v]$
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 $[(\lambda^2 (\lambda x y. \varphi (x^P) (y^P)), x^P, y^P) \equiv (\lambda^2 (\lambda x y. \psi (x^P) (y^P)), x^P, y^P)]$ in v
 using *assms beta-equiv-eq-1-2* by *auto*
 qed

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

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

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

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

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

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

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

lemma *prop-equiv*[*PLM*]:
 shows $[(\forall x. (\{x^P, F\} \equiv \{x^P, G\})) \rightarrow F = G] in v$
 proof (*rule CP*)
 assume $1: [\forall x. \{x^P, F\} \equiv \{x^P, G\}] in v$
 {
 fix x
 have $[\{x^P, F\} \equiv \{x^P, G\}] in v$
 using 1 by (*rule* $\forall E$)
 hence $[\Box(\{x^P, F\} \equiv \{x^P, G\})] in v$

```

    using PLM.en-eq-6  $\equiv E(1)$  by blast
  }
  hence  $[\forall x. \Box(\llbracket x^P, F \rrbracket \equiv \llbracket x^P, G \rrbracket) \text{ in } v]$ 
    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[axiom-instance]
      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

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

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

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

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

lemma thm-relation-negation-1-3[PLM]:
   $[\llbracket F^-, x^P, y^P, z^P \rrbracket \equiv \neg \llbracket F, x^P, y^P, z^P \rrbracket \text{ in } v]$ 

```

unfolding *propnot-defs*
apply (*rule lambda-predicates-2-3*[*axiom-instance*])
by *show-proper*

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 – **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 – **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 – **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 – **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]:
 $[(p::o)^-] = \neg p$ in v
unfolding *propnot-defs* **using** *propositions-lemma-1* **by** *simp*

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

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

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

lemma *thm-cont-prop-1*[PLM]:
 $[NonContingent (F::\Pi_1) \equiv NonContingent (F^-)]$ in v
proof ($rule \equiv I$; $rule CP$)
assume $[NonContingent F]$ in v
hence $[\Box(\forall x. \Box(F, x^P)) \vee \Box(\forall x. \neg \Box(F, x^P))]$ in v
unfolding *NonContingent-def Necessary-defs Impossible-defs* .
hence $[\Box(\forall x. \neg \Box(F^-, x^P)) \vee \Box(\forall x. \neg \Box(F, x^P))]$ in v
apply –
apply (PLM -subst-method $\lambda x . \Box(F, x^P) \lambda x . \neg \Box(F^-, x^P)$)
using *thm-relation-negation-2-1*[*equiv-sym*] **by** *auto*
hence $[\Box(\forall x. \neg \Box(F^-, x^P)) \vee \Box(\forall x. \Box(F^-, x^P))]$ in v
apply –
apply (PLM -subst-goal-method
 $\lambda \varphi . \Box(\forall x. \neg \Box(F^-, x^P)) \vee \Box(\forall x. \varphi x) \lambda x . \neg \Box(F, x^P)$)
using *thm-relation-negation-1-1*[*equiv-sym*] **by** *auto*
hence $[\Box(\forall x. \Box(F^-, x^P)) \vee \Box(\forall x. \neg \Box(F^-, x^P))]$ in v
by ($rule$ *oth-class-taut-3-e*[*equiv-lr*])
thus $[NonContingent (F^-)]$ in v
unfolding *NonContingent-def Necessary-defs Impossible-defs* .
next
assume $[NonContingent (F^-)]$ in v
hence $[\Box(\forall x. \neg \Box(F^-, x^P)) \vee \Box(\forall x. \Box(F^-, x^P))]$ in v
unfolding *NonContingent-def Necessary-defs Impossible-defs*
by ($rule$ *oth-class-taut-3-e*[*equiv-lr*])
hence $[\Box(\forall x. \Box(F, x^P)) \vee \Box(\forall x. \neg \Box(F^-, x^P))]$ in v
apply –
apply (PLM -subst-method $\lambda x . \neg \Box(F^-, x^P) \lambda x . \Box(F, x^P)$)
using *thm-relation-negation-2-1* **by** *auto*
hence $[\Box(\forall x. \Box(F, x^P)) \vee \Box(\forall x. \neg \Box(F, x^P))]$ in v
apply –
apply (PLM -subst-method $\lambda x . \Box(F^-, x^P) \lambda x . \neg \Box(F, x^P)$)
using *thm-relation-negation-1-1* **by** *auto*
thus $[NonContingent F]$ in v
unfolding *NonContingent-def Necessary-defs Impossible-defs* .
qed

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

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

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

lemma *lem-cont-e-2*[PLM]:


```

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

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

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

lemma thm-cont-e-3[PLM]:
[Contingent ( $E!^\neg$ ) 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  $\alpha=E!^\neg$  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  $L \equiv (\lambda x . (\Downarrow E!, x^P) \rightarrow (\Downarrow E!, x^P))$ 

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

lemma thm-noncont-e-e-2[PLM]:
[Impossible ( $L^\neg$ ) 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 show-proper
using if-p-then-p .

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

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

lemma thm-noncont-e-e-5[PLM]:
[ $\exists (F::\Pi_1) G . F \neq G \ \& \ \text{NonContingent } F \ \& \ \text{NonContingent } G$ ] in v]
apply (rule-tac  $\alpha=L$  in  $\exists I$ , rule-tac  $\alpha=L^\neg$  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 – **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 – **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  $\exists G . \text{NonContingent } G \ \& \ G = (E!)^- \text{ in } v$ 
    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  $[\text{NonContingent } (L^-) \ \& \ (L^-) = E! \text{ in } v]$ 
  using thm-noncont-e-e-4 & I by auto
  hence  $\exists G . \text{NonContingent } G \ \& \ G = E! \text{ in } v$ 
  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  $[\text{NonContingent } (L^-) \ \& \ (L^-) = (E!)^- \text{ in } v]$ 
  using thm-noncont-e-e-4 & I by auto
  hence  $\exists G . \text{NonContingent } G \ \& \ G = (E!)^- \text{ in } v$ 
  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]:
 $[\text{NonContingent } (p::o) \equiv \text{NonContingent } (p^-) \text{ in } v]$ 
proof (rule  $\equiv I$ ; rule CP)
  assume  $[\text{NonContingent } p \text{ in } v]$ 
  hence  $[\Box p \vee \Box \neg p \text{ in } v]$ 
  unfolding NonContingent-def Necessary-defs Impossible-defs .
  hence  $[\Box(\neg(p^-)) \vee \Box(\neg p) \text{ in } v]$ 
  apply -
  apply (PLM-subst-method  $p \neg(p^-)$ )
  using thm-relation-negation-4[equiv-sym] by auto
  hence  $[\Box(\neg(p^-)) \vee \Box(p^-) \text{ in } v]$ 
  apply -
  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  $[\Box(p^-) \vee \Box(\neg(p^-)) \text{ in } v]$ 
  by (rule oth-class-taut-3-e[equiv-lr])
  thus  $[\text{NonContingent } (p^-) \text{ in } v]$ 
  unfolding NonContingent-def Necessary-defs Impossible-defs .
next
  assume  $[\text{NonContingent } (p^-) \text{ in } v]$ 
  hence  $[\Box(\neg(p^-)) \vee \Box(p^-) \text{ in } v]$ 
  unfolding NonContingent-def Necessary-defs Impossible-defs
  by (rule oth-class-taut-3-e[equiv-lr])
  hence  $[\Box(p) \vee \Box(p^-) \text{ in } v]$ 
  apply -
  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 -
  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 - apply PLM-solver
  apply (PLM-subst-method  $\neg\neg p$ )
  using oth-class-taut-4-b[equiv-sym] by auto
next
  assume  $(\Diamond p) \ \& \ (\Diamond\neg(p))$  in v
  hence  $(\Diamond\neg(\neg p)) \ \& \ (\Diamond\neg(p))$  in v
  apply - apply PLM-solver
  apply (PLM-subst-method  $p \neg\neg p$ )
  using oth-class-taut-4-b by auto
  hence  $(\neg\Box p) \ \& \ (\neg\Box(\neg p))$  in v
  using KBasic2-2[equiv-rl] &I &E by meson
  hence  $\neg(\Box(p) \vee \Box(\neg p))$  in v
  by (rule oth-class-taut-6-d[equiv-rl])
  thus [Contingent p in v]
  unfolding Contingent-def Necessary-defs Impossible-defs .
qed

```

```

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

```

context

begin

```

private definition p0 where
  p0  $\equiv \forall x. (\Box E!, x^P) \rightarrow (\Box E!, x^P)$ 

```

```

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

```

```

lemma thm-noncont-propos-2[PLM]:
  [Impossible (p0-) in v]
  unfolding Impossible-defs
  apply (PLM-subst-method  $\neg p_0 \ p_0^-$ )
  using thm-relation-negation-3[equiv-sym] apply simp
  apply (PLM-subst-method  $p_0 \neg\neg p_0$ )
  using oth-class-taut-4-b apply simp
  using thm-noncont-propos-1 unfolding Necessary-defs
  by simp

```

```

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

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

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

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

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

lemma basic-prop-2[PLM]:
  [Contingent  $q_0$  in  $v$ ]
  unfolding Contingent-def Necessary-defs Impossible-defs
  apply (rule oth-class-taut-6-d[equiv-rl])
  apply (PLM-subst-goal-method  $\lambda \varphi . (\neg\Box(\varphi)) \ \& \ \neg\Box\neg q_0 \ \neg\neg q_0$ )
  using oth-class-taut-4-b[equiv-sym] apply simp
  using qml-4[axiom-instance, conj-sym]
  unfolding  $q_0\text{-def}$  diamond-def by simp

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

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

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

```

lemma *four-distinct-props-2*[PLM]:
 $[Contingent(p::o) \rightarrow \neg(\exists q. (NonContingent\ q \ \& \ q = p)) \text{ in } v]$
proof (rule CP)
 assume $[Contingent\ p \text{ in } v]$
 hence $[\neg(NonContingent\ p) \text{ in } v]$
 unfolding *NonContingent-def Contingent-def*
 apply – **by** *PLM-solver*
 moreover {
 assume $[\exists q. NonContingent\ q \ \& \ q = p \text{ in } v]$
 then obtain r **where** $[NonContingent\ r \ \& \ r = p \text{ in } v]$
 by (rule $\exists E$)
 hence $[NonContingent\ p \text{ in } v]$
 using *&E l-identity*[*axiom-instance, deduction, deduction*]
 by *blast*
 }
 ultimately show $[\neg(\exists q. NonContingent\ q \ \& \ q = p) \text{ in } v]$
 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 $[\exists q. NonContingent\ q \ \& \ q = q_0 \text{ in } v]$
 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 $[\exists q. NonContingent\ q \ \& \ q = (q_0^-) \text{ in } v]$
 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 $[\exists q. NonContingent\ q \ \& \ q = q_0 \text{ in } v]$
 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 $[\exists q. NonContingent\ q \ \& \ q = (q_0^-) \text{ in } v]$
 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-3]
      modus-tollens-1 CP
    by blast
  next
    show  $[q_0 \neq (q_0^-) \text{ in } v]$ 
    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  $T\Diamond$ [deduction] apply simp
  by (rule &E(2))

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

lemma cont-true-cont-3[PLM]:
  [ContingentlyTrue  $p \equiv$  ContingentlyFalse  $(p^-)$  in  $v$ ]
  unfolding ContingentlyTrue-def ContingentlyFalse-def
  apply (PLM-subst-method  $\neg p^-$ )
  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  $\neg p^-$ )
  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  $[q_0 \vee \neg q_0 \text{ in } v]$ 
    by PLM-solver
    moreover {
      assume  $[q_0 \text{ in } v]$ 
      hence  $[q_0 \ \& \ \Diamond\neg q_0 \text{ in } v]$ 
      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  $q_0$ -def
      using qml-4[axiom-instance,conj1] &I
      by auto
    }
  }
  ultimately show ?thesis
  unfolding ContingentlyTrue-def ContingentlyFalse-def
  using  $\vee E(4)$  CP by auto

```

```

qed

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

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

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

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

lemma oa-contingent-1[PLM]:
  [O! ≠ A! in v]
proof –
  {
    assume [O! = A! in v]
    hence [(λx. ◇(E!, xP)) = (λx. ¬◇(E!, xP)) in v]
      unfolding Ordinary-def Abstract-def .
    moreover have [(λx. ◇(E!, xP)), xP] ≡ ◇(E!, xP) in v]
      apply (rule beta-C-meta-1)
      by show-proper
    ultimately have [(λx. ¬◇(E!, xP)), xP] ≡ ◇(E!, xP) in v]
      using l-identity[axiom-instance, deduction, deduction] by fast
    moreover have [(λx. ¬◇(E!, xP)), xP] ≡ ¬◇(E!, xP) in v]
      apply (rule beta-C-meta-1)
      by show-proper
    ultimately have [◇(E!, xP) ≡ ¬◇(E!, xP) in v]
      apply – by PLM-solver
  }
  thus ?thesis
    using oth-class-taut-1-b modus-tollens-1 CP
    by blast
qed

lemma oa-contingent-2[PLM]:
  [(λO!, xP) ≡ ¬(λA!, xP) in v]
proof –
  have [(λx. ¬◇(E!, xP)), xP] ≡ ¬◇(E!, xP) in v]
    apply (rule beta-C-meta-1)
    by show-proper
  hence [(¬(λx. ¬◇(E!, xP)), xP) ≡ ◇(E!, xP) in v]
    using oth-class-taut-5-d[equiv-lr] oth-class-taut-4-b[equiv-sym]
    ≡ E(5) by blast
  moreover have [(λx. ◇(E!, xP)), xP] ≡ ◇(E!, xP) in v]
    apply (rule beta-C-meta-1)
    by show-proper
  ultimately show ?thesis
    unfolding Ordinary-def Abstract-def
    apply – by PLM-solver
qed

```

```

lemma oa-contingent-3[PLM]:
  [ $\langle A!, x^P \rangle \equiv \neg \langle O!, x^P \rangle$  in  $v$ ]
  using oa-contingent-2
  apply – by PLM-solver

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

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

lemma oa-contingent-6[PLM]:
  [ $\langle O!^- \rangle \neq \langle A!^- \rangle$  in  $v$ ]
  proof –
  {
    assume [ $\langle O!^- \rangle = \langle A!^- \rangle$  in  $v$ ]
    hence [ $\langle \lambda x . \neg \langle O!, x^P \rangle \rangle = \langle \lambda x . \neg \langle A!, x^P \rangle \rangle$  in  $v$ ]
    unfolding propnot-defs .
    moreover have [ $\langle \lambda x . \neg \langle O!, x^P \rangle \rangle, x^P \rangle \equiv \neg \langle O!, x^P \rangle$  in  $v$ ]
    apply (rule beta-C-meta-1)
    by show-proper
    ultimately have [ $\langle \lambda x . \neg \langle A!, x^P \rangle, x^P \rangle \equiv \neg \langle O!, x^P \rangle$  in  $v$ ]
    using l-identity[axiom-instance, deduction, deduction]
    by fast
    hence [ $\langle \neg \langle A!, x^P \rangle \rangle \equiv \neg \langle O!, x^P \rangle$  in  $v$ ]
    apply –
    apply (PLM-subst-method [ $\lambda x . \neg \langle A!, x^P \rangle, x^P \rangle (\neg \langle A!, x^P \rangle)$ )
    apply (safe intro!: beta-C-meta-1)
    by show-proper
    hence [ $\langle O!, x^P \rangle \equiv \neg \langle O!, x^P \rangle$  in  $v$ ]
    using oa-contingent-2 apply – by PLM-solver
  }
  thus ?thesis
  using oth-class-taut-1-b modus-tollens-1 CP

```

by blast
qed

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

lemma *oa-contingent-8*[PLM]:
 $[Contingent (O!^{\neg})]$ in v
 using oa-contingent-4 thm-cont-prop-3[equiv-lr] by auto

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

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

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

```

    apply (PLM-subst-method  $\neg\Diamond(\langle E!, x^P \rangle) (\lambda x. \neg\Diamond(\langle E!, x^P \rangle, x^P))$ )
    apply (safe intro!: beta-C-meta-1[equiv-sym])
  by show-proper
qed

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

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

lemma oa-facts-5[PLM]:
   $[\Diamond(\langle O!, x^P \rangle) \equiv \Box(\langle O!, x^P \rangle) \text{ in } v]$ 
  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]:
   $[\Diamond(\langle A!, x^P \rangle) \equiv \Box(\langle A!, x^P \rangle) \text{ in } v]$ 
  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]:
   $[(\langle O!, x^P \rangle) \equiv \mathcal{A}(\langle O!, x^P \rangle) \text{ in } v]$ 
  apply (rule  $\equiv I$ ; rule CP)
  apply (rule nec-imp-act[deduction, OF oa-facts-1[deduction]]; assumption)
proof -
  assume  $[\mathcal{A}(\langle O!, x^P \rangle) \text{ in } v]$ 
  hence  $[\mathcal{A}(\Diamond(\langle E!, x^P \rangle)) \text{ in } v]$ 
    unfolding Ordinary-def apply -
    apply (PLM-subst-method  $(\lambda x. \Diamond(\langle E!, x^P \rangle, x^P) \Diamond(\langle E!, x^P \rangle))$ )
    apply (safe intro!: beta-C-meta-1)
    by show-proper
  hence  $[\Diamond(\langle E!, x^P \rangle) \text{ in } v]$ 
    using Act-Basic-6[equiv-rl] by auto
  thus  $[(\langle O!, x^P \rangle) \text{ in } v]$ 
    unfolding Ordinary-def apply -
    apply (PLM-subst-method  $\Diamond(\langle E!, x^P \rangle) (\lambda x. \Diamond(\langle E!, x^P \rangle, x^P))$ )
    apply (safe intro!: beta-C-meta-1[equiv-sym])
    by show-proper
qed

lemma oa-facts-8[PLM]:
   $[(\langle A!, x^P \rangle) \equiv \mathcal{A}(\langle A!, x^P \rangle) \text{ in } v]$ 
  apply (rule  $\equiv I$ ; rule CP)
  apply (rule nec-imp-act[deduction, OF oa-facts-2[deduction]]; assumption)
proof -
  assume  $[\mathcal{A}(\langle A!, x^P \rangle) \text{ in } v]$ 
  hence  $[\mathcal{A}(\neg\Diamond(\langle E!, x^P \rangle)) \text{ in } v]$ 
    unfolding Abstract-def apply -
    apply (PLM-subst-method  $(\lambda x. \neg\Diamond(\langle E!, x^P \rangle, x^P) \neg\Diamond(\langle E!, x^P \rangle))$ )
    apply (safe intro!: beta-C-meta-1)
    by show-proper
  hence  $[\mathcal{A}(\Box\neg(\langle E!, x^P \rangle)) \text{ in } v]$ 
    apply -
    apply (PLM-subst-method  $(\neg\Diamond(\langle E!, x^P \rangle)) (\Box\neg(\langle E!, x^P \rangle))$ )
    using KBasic2-4[equiv-sym] by auto
  hence  $[\neg\Diamond(\langle E!, x^P \rangle) \text{ in } v]$ 
    using qml-act-2[axiom-instance, equiv-rl] KBasic2-4[equiv-lr] by auto
  thus  $[(\langle A!, x^P \rangle) \text{ in } v]$ 

```

unfolding *Abstract-def* **apply** –
 apply (*PLM-subst-method* $\neg \Diamond \langle F^-, x^P \rangle \langle \lambda x. \neg \Diamond \langle F^-, x^P \rangle, x^P \rangle$)
 apply (*safe intro!*: *beta-C-meta-1*[*equiv-sym*])
 by *show-proper*
 qed

lemma *cont-nec-fact1-1*[*PLM*]:

[*WeaklyContingent* $F \equiv \text{WeaklyContingent } (F^-) \text{ in } v$]

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

assume [*WeaklyContingent* F in v]

hence *wc-def*: [*Contingent* $F \ \& \ (\forall x. (\Diamond \langle F, x^P \rangle \rightarrow \Box \langle F, x^P \rangle))$ in v]

unfolding *WeaklyContingent-def* .

have [*Contingent* (F^-) in v]

using *wc-def*[*conj1*] **by** (*rule* *thm-cont-prop-3*[*equiv-lr*])

moreover {

{

fix x

assume [$\Diamond \langle F^-, x^P \rangle$ in v]

hence [$\neg \Box \langle F, x^P \rangle$ in v]

unfolding *diamond-def* **apply** –

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 [$\Diamond \langle F, x^P \rangle$ in v]

unfolding *diamond-def*

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

apply (*safe intro!*: *beta-C-meta-1*)

by *show-proper*

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

hence [$\neg \Box \langle F^-, x^P \rangle$ in v]

unfolding *diamond-def* **apply** –

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

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

moreover {

assume [$\neg \Box \langle F, x^P \rangle$ in v]

hence [$\Diamond \langle F^-, x^P \rangle$ in v]

unfolding *diamond-def*

apply – **apply** (*PLM-subst-method* $\langle F, x^P \rangle \neg \langle F^-, x^P \rangle$)

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

hence $\Box(\langle F^-, x^P \rangle \text{ in } v)$
 using *wc-def[conj2]* *cqt-1[axiom-instance, deduction]*
modus-ponens **by** *fast*
 }
 ultimately have $\Box(\langle F, x^P \rangle \text{ in } v)$
 using $\neg\neg E$ *modus-tollens-1 CP* **by** *blast*
 }
 hence $[\forall x . \Diamond(\langle F, x^P \rangle) \rightarrow \Box(\langle F, x^P \rangle) \text{ in } v]$
 using $\forall I$ *CP* **by** *fast*
 }
 ultimately show $[WeaklyContingent(F) \text{ in } v]$
 unfolding *WeaklyContingent-def* **by** (rule $\&I$)
qed

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

lemma *cont-nec-fact2-1[PLM]*:
 $[WeaklyContingent(O!) \text{ in } v]$
 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]*:
 $[WeaklyContingent(A!) \text{ in } v]$
 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]*:
 $[\neg(WeaklyContingent(E!)) \text{ in } v]$
proof (rule *modus-tollens-1, rule CP*)
 assume $[WeaklyContingent E! \text{ in } v]$
 thus $[\forall x . \Diamond(\langle E!, x^P \rangle) \rightarrow \Box(\langle E!, x^P \rangle) \text{ in } v]$
 unfolding *WeaklyContingent-def* using $\&E(2)$ **by** *fast*
next
 {
 assume 1: $[\forall x . \Diamond(\langle E!, x^P \rangle) \rightarrow \Box(\langle E!, x^P \rangle) \text{ in } v]$
 have $[\exists x . \Diamond(\langle E!, x^P \rangle \ \& \ \Diamond(\neg(\langle E!, x^P \rangle))) \text{ in } v]$
 using *qml-4[axiom-instance, conj1, THEN BFs-3[deduction]]* .
 then obtain x where $[\Diamond(\langle E!, x^P \rangle \ \& \ \Diamond(\neg(\langle E!, x^P \rangle))) \text{ in } v]$
 by (rule $\exists E$)
 hence $[\Diamond(\langle E!, x^P \rangle \ \& \ \Diamond(\neg(\langle E!, x^P \rangle))) \text{ in } v]$
 using *KBasic2-8[deduction]* *S5Basic-8[deduction]*
 $\&I$ $\&E$ **by** *blast*
 hence $[\Box(\langle E!, x^P \rangle \ \& \ (\neg\Box(\langle E!, x^P \rangle))) \text{ in } v]$
 using 1[*THEN $\forall E$, deduction*] $\&E$ $\&I$
KBasic2-2[equiv-rl] **by** *blast*
 hence $[\neg(\forall x . \Diamond(\langle E!, x^P \rangle) \rightarrow \Box(\langle E!, x^P \rangle)) \text{ in } v]$
 using *oth-class-taut-1-a modus-tollens-1 CP* **by** *blast*
 }
 thus $[\neg(\forall x . \Diamond(\langle E!, x^P \rangle) \rightarrow \Box(\langle E!, x^P \rangle)) \text{ in } v]$
 using *reductio-aa-2 if-p-then-p CP* **by** *meson*
qed

lemma *cont-nec-fact2-4[PLM]*:
 $[\neg(WeaklyContingent(PLM.L)) \text{ in } v]$

```

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

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

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

lemma id-nec3-1[PLM]:
[((xP) =E (yP)) ≡ (□((xP) =E (yP))) in v]
proof (rule ≡I; rule CP)

```

```

assume  $[(x^P) =_E (y^P) \text{ in } v]$ 
hence  $[(\downarrow O!, x^P) \text{ in } v] \wedge [(\downarrow O!, y^P) \text{ in } v] \wedge [\Box(\forall F. (\downarrow F, x^P) \equiv (\downarrow F, y^P)) \text{ in } v]$ 
  using eq-E-simple-1[equiv-lr] using  $\&E$  by blast
hence  $[\Box(\downarrow O!, x^P) \text{ in } v] \wedge [\Box(\downarrow O!, y^P) \text{ in } v]$ 
   $\wedge [\Box(\Box(\forall F. (\downarrow F, x^P) \equiv (\downarrow F, y^P)) \text{ in } v)]$ 
  using oa-facts-1[deduction] S5Basic-6[deduction] by blast
hence  $[\Box((\downarrow O!, x^P) \& (\downarrow O!, y^P) \& \Box(\forall F. (\downarrow F, x^P) \equiv (\downarrow F, y^P))) \text{ in } v]$ 
  using  $\&I$  KBasic-3[equiv-rl] by presburger
thus  $[\Box((x^P) =_E (y^P)) \text{ in } v]$ 
apply –
apply (PLM-subst-method
   $(\downarrow O!, x^P) \& (\downarrow O!, y^P) \& \Box(\forall F. (\downarrow F, x^P) \equiv (\downarrow F, y^P))$ 
   $(x^P) =_E (y^P))$ 
  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*]:

```

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

```

lemma *id-nec4-1*[*PLM*]:

```

 $[(x^P) \neq_E (y^P) \equiv \Box((x^P) \neq_E (y^P)) \text{ in } v]$ 
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 –
    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 $[\mathcal{A}((x^P) =_E (y^P)) \text{ in } v]$
hence $[\mathcal{A}(\Box O!, x^P) \ \& \ \Box O!, y^P) \ \& \ \Box(\forall F . \Box(F, x^P) \equiv \Box(F, y^P)) \text{ in } v]$
apply –
apply (*PLM-subst-method*
 $(x^P) =_E (y^P)$
 $(\Box O!, x^P) \ \& \ \Box O!, y^P) \ \& \ \Box(\forall F . \Box(F, x^P) \equiv \Box(F, y^P))$)
using *eq-E-simple-1* **by** *auto*
hence $[\mathcal{A}(\Box O!, x^P) \ \& \ \mathcal{A}(\Box O!, y^P) \ \& \ \mathcal{A}(\Box(\forall F . \Box(F, x^P) \equiv \Box(F, y^P)) \text{ in } v]$
using *Act-Basic-2*[equiv-lr] $\&I$ $\&E$ **by** *meson*
thus $[(x^P) =_E (y^P) \text{ in } v]$
apply – **apply** (*rule eq-E-simple-1*[equiv-rl])
using *oa-facts-7*[equiv-rl] *qml-act-2*[axiom-instance, equiv-rl]
 $\&I$ $\&E$ **by** *meson*
qed

lemma *id-act-2*[PLM]:
 $[(x^P) \neq_E (y^P) \equiv (\mathcal{A}((x^P) \neq_E (y^P))) \text{ in } v]$
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*: $[\mathcal{A}(\alpha = \beta) \text{ in } v] \implies [(\alpha = \beta) \text{ in } v]$

instantiation $\nu :: \textit{id-act}$

begin

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

```

      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  $\Pi_1 :: id-act$ 
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  $o :: id-act$ 
begin
  instance proof
    interpret PLM .
    fix  $p :: o$  and  $q :: o$  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  $\Pi_2 :: id-act$ 
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  $[\mathcal{A}((\lambda y. \langle F, x^P, y^P \rangle) = (\lambda y. \langle G, x^P, y^P \rangle))$ 
        &  $(\lambda y. \langle F, y^P, x^P \rangle) = (\lambda y. \langle G, y^P, x^P \rangle) \text{ in } v]$ 
        using a logic-actual-nec-3[axiom-instance, equiv-lr] cqt-basic-4[equiv-lr]  $\forall E$ 
        unfolding identity2-def by fast
      hence  $[(\lambda y. \langle F, x^P, y^P \rangle) = (\lambda y. \langle G, x^P, y^P \rangle)]$ 
        &  $((\lambda y. \langle F, y^P, x^P \rangle) = (\lambda y. \langle G, y^P, x^P \rangle)) \text{ in } v]$ 
        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 [ $\mathcal{A}(\text{?}p \ x \ y \ \text{in } v)$ ]
    using a logic-actual-nec-3[axiom-instance, equiv-lr]
          cqt-basic-4[equiv-lr]  $\forall E$ [where 'a= $\nu$ ]
    unfolding identity3-def by blast
  hence [ $\text{?}p \ x \ y \ \text{in } v$ ]
    using  $\&I$   $\&E$  id-act-prop Act-Basic-2[equiv-lr] by metis
}
hence [ $\forall \ y \ . \ \text{?}p \ x \ y \ \text{in } v$ ]
  by (rule  $\forall I$ )
}
thus [ $F = G \ \text{in } v$ ]
  unfolding identity3-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

lemma *eta-conversion-lemma-1*[*PLM*]:

$[(\lambda \ x \ . \ (F, x^P)) = F \ \text{in } v]$

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*]:

$[(\lambda^2 \ (\lambda \ x \ y \ . \ (F, x^P, y^P))) = F \ \text{in } v]$

using lambda-predicates-3-2[axiom-instance] .

lemma *eta-conversion-lemma-3*[*PLM*]:

$[(\lambda^3 \ (\lambda \ x \ y \ z \ . \ (F, x^P, y^P, z^P))) = F \ \text{in } v]$

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*]:

$[\forall \ x \ . \ (O!, x^P) \vee (A!, x^P) \ \text{in } v]$

```

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  $[\Diamond \langle E!, x^P \rangle \equiv \langle \lambda y . \Diamond \langle E!, y^P \rangle, x^P \rangle \text{ in } v]$ 
    apply (rule beta-C-meta-1[equiv-sym])
    by show-proper
  moreover have  $[(\neg \Diamond \langle E!, x^P \rangle) \equiv \langle \lambda y . \neg \Diamond \langle E!, y^P \rangle, x^P \rangle \text{ in } v]$ 
    apply (rule beta-C-meta-1[equiv-sym])
    by show-proper
  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  $[\langle O!, x \rangle \text{ in } v]$ 
  moreover have  $[\Box(\forall F . \langle F, x \rangle \equiv \langle F, x \rangle) \text{ in } v]$ 
    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  $1: [\langle O!, x \rangle \ \& \ \langle O!, y \rangle \ \& \ \Box(\forall F . \langle F, x \rangle \equiv \langle F, y \rangle) \text{ in } v]$ 
    using eq-E-simple-1[equiv-lr] by simp
  have  $[\Box(\forall F . \langle F, y \rangle \equiv \langle F, x \rangle) \text{ in } v]$ 
    apply (PLM-subst-method
       $\lambda F . \langle F, x \rangle \equiv \langle F, y \rangle$ 
       $\lambda F . \langle F, y \rangle \equiv \langle F, x \rangle$ )
    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  $[\Box(\forall F . \langle F, x \rangle \equiv \langle F, y \rangle) \ \& \ (\forall F . \langle F, y \rangle \equiv \langle F, z \rangle) \text{ in } v]$ 
    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 [( $\forall F . \langle F, x \rangle \equiv \langle F, y \rangle$ ) & ( $\forall F . \langle F, y \rangle \equiv \langle F, z \rangle$ )]
     $\rightarrow (\forall F . \langle F, x \rangle \equiv \langle F, z \rangle)$  in w]
  by PLM-solver
}
hence [ $\Box((\forall F . \langle F, x \rangle \equiv \langle F, y \rangle) \& (\forall F . \langle F, y \rangle \equiv \langle F, z \rangle))$ 
 $\rightarrow (\forall F . \langle F, x \rangle \equiv \langle F, z \rangle)$  in v]
  by (rule RN)
}
ultimately have [ $\Box(\forall F . \langle F, x \rangle \equiv \langle F, z \rangle)$  in v]
  using qml-1[axiom-instance,deduction,deduction] by blast
thus [ $x =_E z$  in v]
  using a[conj1, THEN eq-E-simple-1[equiv-lr,conj1,conj1]]
  using a[conj2, THEN eq-E-simple-1[equiv-lr,conj1,conj2]]
    eq-E-simple-1[equiv-rl] &I
  by presburger
qed

```

lemma *ord-eq-E-eq[PLM]*:

```

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

```

lemma *ord-eq-E[PLM]*:

```

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

```

lemma *ord-eq-E2*[PLM]:

$$[(\langle O!, x^P \rangle \ \& \ \langle O!, y^P \rangle) \rightarrow ((x^P \neq y^P) \equiv (\lambda z . z^P =_E x^P) \neq (\lambda z . z^P =_E y^P)) \text{ in } v]$$

proof (*rule CP*; *rule $\equiv I$* ; *rule CP*)
assume *ord-xy*: $[(\langle O!, x^P \rangle \ \& \ \langle O!, y^P \rangle) \text{ in } v]$
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 *identity_E-infix-def*
apply *show-proper*
using *ord-eq-Eequiv-1*[*deduction*] *ord-xy*[*conjI*]
unfolding *identity_E-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 $[x^P =_E y^P \text{ in } v]$
unfolding *identity_E-infix-def*
apply (*safe intro!*:
beta-C-meta-1[**where** $\varphi = \lambda z . (\langle \text{basic-identity}_{E,z,y^P} \rangle, \text{equiv-lr})$]
by *show-proper*
}
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*: $[(\langle O!, x^P \rangle \ \& \ \langle O!, y^P \rangle) \text{ in } v]$
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]:

$$[(\langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle) \rightarrow ((\forall F . \langle x^P, F \rangle \equiv \langle y^P, F \rangle) \rightarrow x^P = y^P) \text{ in } v]$$

proof(*rule CP*; *rule CP*)
assume *abs-xy*: $[(\langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle) \text{ in } v]$
assume *enc-equiv*: $[\forall F . \langle x^P, F \rangle \equiv \langle y^P, F \rangle \text{ in } v]$
{
fix *P*
have $[\langle x^P, P \rangle \equiv \langle y^P, P \rangle \text{ in } v]$
using *enc-equiv* **by** (*rule $\forall E$*)
hence $[\Box(\langle x^P, P \rangle \equiv \langle y^P, P \rangle) \text{ in } v]$
using *en-eq-2 intro-elim-6-e intro-elim-6-f*
en-eq-5[*equiv-rl*] **by** *meson*
}
hence $[\Box(\forall F . \langle x^P, F \rangle \equiv \langle y^P, F \rangle) \text{ in } v]$
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]:

$$[(\langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle) \rightarrow ((\exists F . \langle x^P, F \rangle \ \& \ \neg \langle y^P, F \rangle) \rightarrow x^P \neq y^P) \text{ in } v]$$

proof(*rule CP*; *rule CP*)

```

assume abs-xy: [ $\langle A!, x^P \rangle$  &  $\langle A!, y^P \rangle$  in  $v$ ]
assume [ $\exists F . \langle x^P, F \rangle$  &  $\neg \langle y^P, F \rangle$  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 ( $PLM$ -subst-method  $\lambda x . \Diamond \langle E!, x^P \rangle \ \lambda x . \langle \lambda y . \Diamond \langle E!, y^P \rangle, x^P \rangle$ )
  apply (safe intro!: beta-C-meta-1[equiv-sym])
  apply show-proper
  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 ( $PLM$ -subst-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[ $PLM$ ]:
  [ $\Box(\exists x . \langle A!, x^P \rangle)$  in  $v$ ]
  proof –
    {
      fix  $v$ 
      have [ $\exists x . \langle A!, x^P \rangle \ \& \ (\forall F . \langle x^P, F \rangle \equiv (F = F))$  in  $v$ ]
        using A-objects[axiom-instance] by simp
    }

```

hence $[\exists x . \langle A!, x^P \rangle \text{ in } v]$
 using *cqt-further-5*[*deduction, conj1*] **by** *fast*
 }
 thus *?thesis* **by** (*rule RN*)
qed

lemma *a-objects-exist-2*[*PLM*]:
 $[\Box(\neg(\forall x . \langle O!, x^P \rangle)) \text{ in } v]$
apply (*PLM-subst-method* $(\exists x . \neg \langle O!, x^P \rangle) \neg(\forall x . \langle O!, x^P \rangle)$)
 using *cqt-further-2*[*equiv-sym*] **apply** *fast*
apply (*PLM-subst-method* $\lambda x . \langle A!, x^P \rangle \lambda x . \neg \langle O!, x^P \rangle$)
 using *oa-contingent-3* *a-objects-exist-1* **by** *auto*

lemma *a-objects-exist-3*[*PLM*]:
 $[\Box(\neg(\forall x . \langle E!, x^P \rangle)) \text{ in } v]$
proof –
 {
 fix *v*
 have $[\exists x . \langle A!, x^P \rangle \ \& \ (\forall F . \langle x^P, F \rangle \equiv (F = F)) \text{ in } v]$
 using *A-objects*[*axiom-instance*] **by** *simp*
 hence $[\exists x . \langle A!, x^P \rangle \text{ in } v]$
 using *cqt-further-5*[*deduction, conj1*] **by** *fast*
 then obtain *a* where
 $[\langle A!, a^P \rangle \text{ in } v]$
by (*rule* $\exists E$)
 hence $[\neg(\langle E!, a^P \rangle) \text{ in } v]$
 unfolding *Abstract-def*
apply (*safe intro!*: *beta-C-meta-1*[*equiv-lr*])
by *show-proper*
 hence $[(\neg \langle E!, a^P \rangle) \text{ in } v]$
 using *KBasic2-4*[*equiv-rl*] *qml-2*[*axiom-instance, deduction*]
by *simp*
 hence $[\neg(\forall x . \langle E!, x^P \rangle) \text{ in } v]$
 using $\exists I$ *cqt-further-2*[*equiv-rl*]
by *fast*
 }
 thus *?thesis*
by (*rule RN*)
qed

lemma *encoders-are-abstract*[*PLM*]:
 $[(\exists F . \langle x^P, F \rangle) \rightarrow \langle A!, x^P \rangle \text{ in } v]$
 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*]:
 $[\exists! x . \langle A!, x^P \rangle \ \& \ (\forall F . \langle x^P, F \rangle \equiv \varphi F) \text{ in } v]$
proof –
 have $[\exists x . \langle A!, x^P \rangle \ \& \ (\forall F . \langle x^P, F \rangle \equiv \varphi F) \text{ in } v]$
 using *A-objects*[*axiom-instance*] **by** *simp*
 then obtain *a* where *a-prop*:
 $[\langle A!, a^P \rangle \ \& \ (\forall F . \langle a^P, F \rangle \equiv \varphi F) \text{ in } v]$ **by** (*rule* $\exists E$)
moreover have $[\forall y . \langle A!, y^P \rangle \ \& \ (\forall F . \langle y^P, F \rangle \equiv \varphi F) \rightarrow (y = a) \text{ in } v]$
proof (*rule* $\forall I$; *rule CP*)
 fix *b*
 assume *b-prop*: $[\langle A!, b^P \rangle \ \& \ (\forall F . \langle b^P, F \rangle \equiv \varphi F) \text{ in } v]$
 {
 fix *P*
 have $[\langle b^P, P \rangle \equiv \langle a^P, P \rangle \text{ in } v]$
 using *a-prop*[*conj2*] *b-prop*[*conj2*] $\equiv I \equiv E(1) \equiv E(2)$
CP *vdash-properties-10* $\forall E$ **by** *metis*


```

}
hence  $[\forall F . \llbracket b^P, F \rrbracket \equiv \llbracket a^P, F \rrbracket \text{ in } v]$ 
  using  $\forall I$  by fast
thus  $[b = a \text{ in } v]$ 
  unfolding identity- $\nu$ -def
  using ab-obey-1[deduction, deduction]
        a-prop[conj1] b-prop[conj1] &I by blast
qed
ultimately show ?thesis
  unfolding exists-unique-def
  using &I  $\exists I$  by fast
qed

```

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

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

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

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

lemma *obj-oth-5*[PLM]:
 $[\exists! x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv (F = G)) \text{ in } v]$
 using *A-objects-unique* .

lemma *obj-oth-6*[PLM]:
 $[\exists! x . \llbracket A!, x^P \rrbracket \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \Box(\forall y . \llbracket G, y^P \rrbracket \rightarrow \llbracket F, y^P \rrbracket)) \text{ in } v]$
 using *A-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 $\mathcal{A}(\varphi \alpha \rightarrow \beta = \alpha)$
 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  $\alpha$  where 1:
   $[\mathcal{A}\varphi \alpha \ \& \ (\forall \beta. \mathcal{A}\varphi \beta \rightarrow \beta = \alpha) \text{ in } v]$ 
  by (rule  $\exists E$ )
{
  fix  $\beta$ 
  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

```

lemma desc-nec-encode[PLM]:
  [ $\llbracket \iota x . \langle A!, x^P \rangle \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi F), G \rrbracket \equiv \mathcal{A}(\varphi G) \text{ in } v$ ]
proof –
  obtain  $a$  where
    [ $a^P = (\iota x . \langle A!, x^P \rangle \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv \varphi F)) \text{ in } v$ ]
    using  $A$ -descriptions by (rule  $\exists E$ )
  moreover {
    hence [ $\mathcal{A}(\langle A!, a^P \rangle \ \& \ (\forall F . \llbracket a^P, F \rrbracket \equiv \varphi F)) \text{ in } v$ ]
      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  $?x = \iota x . \langle A!, x^P \rangle \ \& \ (\forall F . \llbracket x^P, F \rrbracket \equiv (\exists q . q \ \& \ F = (\lambda y . q)))$ 
  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 [ $\llbracket ?x, \lambda y . p_1 \rrbracket \equiv \mathcal{A}(\exists q . q \ \& \ (\lambda y . p_1) = (\lambda y . q)) \text{ in } v$ ]
    using desc-nec-encode by fast
  ultimately have [ $\llbracket ?x, \lambda y . p_1 \rrbracket \text{ in } v$ ]
    using  $\equiv E$  by blast
end

```

```

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

```

qed

lemma *Box-desc-encode-2*[PLM]:

$[\Box(\varphi \ G) \rightarrow \Box(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)), G \rangle \equiv \varphi \ G)$ in v

proof (rule CP)

assume a : $[\Box(\varphi \ G) \text{ in } v]$

hence $[\Box(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)), G \rangle \rightarrow \varphi \ G)$ in v

using *KBasic-1*[deduction] by simp

moreover {

have $[\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)), G \rangle$ in v

using *a Box-desc-encode-1*[deduction] by auto

hence $[\Box(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)), G \rangle$ in v

using *encoding*[axiom-instance, deduction] by blast

hence $[\Box(\varphi \ G \rightarrow \langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)), G \rangle$ in v

using *KBasic-1*[deduction] by simp

}

ultimately show $[\Box(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)), G \rangle \equiv \varphi \ G)$ in v

using $\&I$ *KBasic-4*[equiv-rl] by blast

qed

lemma *box-phi-a-1*[PLM]:

assumes $[\Box(\forall \ F \ . \ \varphi \ F \rightarrow \Box(\varphi \ F)) \text{ in } v]$

shows $[(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)) \rightarrow \Box(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)) \text{ in } v]$

proof (rule CP)

assume a : $[(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)) \text{ in } v]$

have $[\Box(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)) \text{ in } v]$

using *oa-facts-2*[deduction] *a*[conj1] by auto

moreover have $[\Box(\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F) \text{ in } v]$

proof (rule BF[deduction]; rule $\forall I$)

fix F

have ϑ : $[\Box(\varphi \ F \rightarrow \Box(\varphi \ F)) \text{ in } v]$

using *assms*[THEN CBF[deduction]] by (rule $\forall E$)

moreover have $[\Box(\langle x^P , F \rangle \rightarrow \Box(\langle x^P , F \rangle)) \text{ in } v]$

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] $\forall E$ [where $'a=\Pi_1] \equiv E$ by blast

thus $[\Box(\varphi \ F) \text{ in } v]$

using ϑ [THEN *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] $\forall E$ [where $'a=\Pi_1] \equiv E$ 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 $[\Box(\langle \mathcal{A}! , x^P \rangle \ \& \ (\forall \ F \ . \ \langle x^P , F \rangle \equiv \varphi \ F)) \text{ in } v]$

using $\&I$ *KBasic-3*[equiv-rl] by blast

qed

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 $[\langle a^P, G \rangle \equiv \varphi G \text{ in } v] \text{ by (rule } \forall E)$
}
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 $[\exists x . \langle A!, x^P \rangle \ \& \ (\forall F . \langle x^P, F \rangle \equiv (F \neq F)) \text{ in } v]$
using *A-objects*[*axiom-instance*] **by** *simp*
then obtain *a* **where** *a-prop*:
 $[\langle A!, a^P \rangle \ \& \ (\forall F . \langle a^P, F \rangle \equiv (F \neq F)) \text{ in } v]$
by (rule $\exists E$)
have *1*: $[\langle A!, a^P \rangle \ \& \ (\neg(\exists F . \langle a^P, F \rangle)) \text{ in } v]$
using *a-prop*[*conj1*] **apply** (rule $\&I$)
proof –
{
assume $[\exists F . \langle a^P, F \rangle \text{ in } v]$
then obtain *P* **where**
 $[\langle a^P, P \rangle \text{ in } v] \text{ by (rule } \exists E)$
hence $[P \neq P \text{ in } v]$
using *a-prop*[*conj2, THEN $\forall E$, equiv-lr*] **by** *simp*
hence $[\neg(\exists F . \langle a^P, F \rangle) \text{ in } v]$
using *id-eq-1 reductio-aa-1* **by** *fast*
}
thus $[\neg(\exists F . \langle a^P, F \rangle) \text{ in } v]$
using *reductio-aa-1* **by** *blast*
qed
moreover have $[\forall y . ((\langle A!, y^P \rangle) \ \& \ (\neg(\exists F . \langle y^P, F \rangle))) \rightarrow y = a \text{ in } v]$
proof (rule $\forall I$; rule *CP*)
fix *y*
assume *2*: $[(\langle A!, y^P \rangle) \ \& \ (\neg(\exists F . \langle y^P, F \rangle)) \text{ in } v]$
have $[\forall F . \langle y^P, F \rangle \equiv \langle a^P, F \rangle \text{ in } v]$
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: $[(\llbracket A!, a^P \rrbracket) \ \& \ (\forall F . \llbracket a^P, F \rrbracket) \text{ in } v]$
 using *a-prop*[*conj1*] apply (*rule* $\&I$)
 using $\forall I$ *a-prop*[*conj2*, *THEN* $\forall E$, *equiv-rl*] *id-eq-1* by fast
 moreover have $[\forall y . ((\llbracket A!, y^P \rrbracket) \ \& \ (\forall F . \llbracket y^P, F \rrbracket)) \rightarrow y = a \text{ in } v]$
 proof (*rule* $\forall I$; *rule* *CP*)
 fix *y*
 assume 2: $[(\llbracket A!, y^P \rrbracket) \ \& \ (\forall F . \llbracket y^P, F \rrbracket) \text{ in } v]$
 have $[\forall F . \llbracket y^P, F \rrbracket \equiv \llbracket a^P, F \rrbracket \text{ in } v]$
 using *cqt-further-11*[*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 *Universal-def exists-unique-def* by fast
 qed

lemma *null-uni-uniq-3*[PLM]:

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

using *null-uni-uniq-1*[*THEN RN*, *THEN nec-imp-act*[*deduction*]]
A-Exists-2[*equiv-rl*] by auto

lemma *null-uni-uniq-4*[PLM]:

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

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]:

$[\text{Null } (x^P) \rightarrow \Box(\text{Null } (x^P)) \text{ in } v]$

proof (*rule* *CP*)

assume $[\text{Null } (x^P) \text{ in } v]$
 hence 1: $[(\llbracket A!, x^P \rrbracket) \ \& \ (\neg(\exists F . \llbracket x^P, F \rrbracket)) \text{ in } v]$
 unfolding *Null-def* .
 have $[\Box(\llbracket A!, x^P \rrbracket) \text{ in } v]$
 using 1[*conj1*] *oa-facts-2*[*deduction*] by simp
 moreover have $[\Box(\neg(\exists F . \llbracket x^P, F \rrbracket)) \text{ in } v]$
 proof –
 {
 assume $[\neg\Box(\neg(\exists F . \llbracket x^P, F \rrbracket)) \text{ in } v]$
 hence $[\Diamond(\exists F . \llbracket x^P, F \rrbracket) \text{ in } v]$
 unfolding *diamond-def* .
 hence $[\exists F . \Diamond\llbracket x^P, F \rrbracket \text{ in } v]$
 using *BF \Diamond* [*deduction*] by blast
 then obtain *P* where $[\Diamond\llbracket x^P, P \rrbracket \text{ in } v]$
 by (*rule* $\exists E$)
 hence $[\llbracket x^P, P \rrbracket \text{ in } v]$
 using *en-eq-3*[*equiv-lr*] by simp
 hence $[\exists F . \llbracket x^P, F \rrbracket \text{ in } v]$
 using $\exists I$ by fast
 }

```

    }
    thus ?thesis
      using 1[conj2] modus-tollens-1 CP
      useful-tautologies-1[deduction] by metis
  qed
ultimately show  $\Box \text{Null} (x^P)$  in v
  unfolding Null-def
  using &I KBasic-3[equiv-rl] by blast
qed

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

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

lemma null-uni-facts-4[PLM]:
  [Universal  $(a_V)$  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 $[? \psi (y^P) \text{ in } v]$
 using 1[*THEN $\forall E$, deduction*]
 unfolding *UniversalObject-def* by *simp*
 ultimately show $[? \psi (a_V) \text{ in } v]$
 using *l-identity*[*axiom-instance, deduction, deduction*] by *blast*
 qed

lemma *aclassical-1*[*PLM*]:

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

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

lemma $a\text{classical}\text{-3}[PLM]$:

$[\forall F . \exists x y . \langle A!, x^P \rangle \& \langle A!, y^P \rangle \& (x \neq y)$
 $\& ((\lambda^0 \langle F, x^P \rangle) = (\lambda^0 \langle F, y^P \rangle))$ in $v]$

proof (rule $\forall I$)

fix R

obtain a **where** ϑ :

$\langle \langle A!, a^P \rangle \& (\forall F . \langle a^P, F \rangle \equiv (\exists y . \langle A!, y^P \rangle$
 $\& F = (\lambda z . \langle R, y^P \rangle) \& \neg \langle y^P, F \rangle))$ in $v]$

using $A\text{-objects}[axiom\text{-instance}]$ **by** (rule $\exists E$)

{
assume $\neg \langle a^P, (\lambda z . \langle R, a^P \rangle) \rangle$ in v
hence $\neg (\langle A!, a^P \rangle \& (\lambda z . \langle R, a^P \rangle) = (\lambda z . \langle R, a^P \rangle))$
 $\& \neg \langle a^P, (\lambda z . \langle R, a^P \rangle) \rangle$ in v
using $\vartheta[conj2, THEN \forall E, THEN oth\text{-class}\text{-taut}\text{-5}\text{-d}[equiv\text{-lr}], equiv\text{-lr}]$
 $cqt\text{-further}\text{-4}[equiv\text{-lr}] \forall E$ **by fast**
hence $\langle \langle A!, a^P \rangle \& (\lambda z . \langle R, a^P \rangle) \rangle = (\lambda z . \langle R, a^P \rangle)$
 $\rightarrow \langle a^P, (\lambda z . \langle R, a^P \rangle) \rangle$ in v
apply – **by PLM-solver**
hence $\langle \langle a^P, (\lambda z . \langle R, a^P \rangle) \rangle \rangle$ in v
using $\vartheta[conj1] id\text{-eq}\text{-1} \& I \vdash\text{-properties}\text{-10}$ **by fast**
}
hence $1: \langle \langle a^P, (\lambda z . \langle R, a^P \rangle) \rangle \rangle$ in v

using *reductio-aa-1 CP if-p-then-p* by *blast*
 then obtain *b* where ξ :
 $[(\lambda! . b^P) \ \& \ (\lambda \ z . (\lambda! . R, a^P))] = (\lambda \ z . (\lambda! . R, b^P))$
 $\& \neg \llbracket b^P, (\lambda \ z . (\lambda! . R, a^P)) \rrbracket$ in *v*
 using $\vartheta[\text{conj2}, \text{THEN } \forall E, \text{equiv-lr}] \exists E$ by *blast*
 have $[a \neq b$ in *v*]
 proof –
 {
 assume $[a = b$ in *v*]
 hence $\llbracket b^P, (\lambda \ z . (\lambda! . R, a^P)) \rrbracket$ 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 $[(\lambda! . R, a^P) = (\lambda! . R, b^P)]$ in *v*
 unfolding *identity_o-def*
 using $\xi[\text{conj1}, \text{conj2}]$ by *auto*
 hence $[(\lambda^0 (\lambda! . R, a^P)) = (\lambda^0 (\lambda! . R, b^P))]$ in *v*
 using *lambda-p-q-p-eq-q[equiv-rl]* by *simp*
 }
 ultimately have $[(\lambda! . a^P) \ \& \ (\lambda! . b^P) \ \& \ a \neq b$
 $\& \ ((\lambda^0 (\lambda! . R, a^P)) = (\lambda^0 (\lambda! . R, b^P)))]$ in *v*
 using $\vartheta[\text{conj1}] \ \xi[\text{conj1}, \text{conj1}] \ \xi[\text{conj1}, \text{conj2}] \ \& I$
 by *presburger*
 hence $[\exists \ y . (\lambda! . a^P) \ \& \ (\lambda! . y^P) \ \& \ a \neq y$
 $\& \ (\lambda^0 (\lambda! . R, a^P)) = (\lambda^0 (\lambda! . R, y^P))]$ in *v*
 using $\exists I$ by *fast*
 thus $[\exists \ x \ y . (\lambda! . x^P) \ \& \ (\lambda! . y^P) \ \& \ x \neq y$
 $\& \ (\lambda^0 (\lambda! . R, x^P)) = (\lambda^0 (\lambda! . R, y^P))]$ in *v*
 using $\exists I$ by *fast*
 qed

lemma *aclassical2[PLM]*:

$[\exists \ x \ y . (\lambda! . x^P) \ \& \ (\lambda! . y^P) \ \& \ x \neq y \ \& \ (\forall \ F . (\lambda! . F, x^P) \equiv (\lambda! . F, y^P))]$ in *v*
 proof –
 let $?R_1 = \lambda^2 (\lambda \ x \ y . \forall \ F . (\lambda! . F, x^P) \equiv (\lambda! . F, y^P))$
 have $[\exists \ x \ y . (\lambda! . x^P) \ \& \ (\lambda! . y^P) \ \& \ x \neq y$
 $\& \ (\lambda z . (\lambda! . ?R_1, z^P, x^P)) = (\lambda z . (\lambda! . ?R_1, z^P, y^P))]$ in *v*
 using *aclassical-1* by (rule $\forall E$)
 then obtain *a* where
 $[\exists \ y . (\lambda! . a^P) \ \& \ (\lambda! . y^P) \ \& \ a \neq y$
 $\& \ (\lambda z . (\lambda! . ?R_1, z^P, a^P)) = (\lambda z . (\lambda! . ?R_1, z^P, y^P))]$ in *v*
 by (rule $\exists E$)
 then obtain *b* where *ab-prop*:
 $[(\lambda! . a^P) \ \& \ (\lambda! . b^P) \ \& \ a \neq b$
 $\& \ (\lambda z . (\lambda! . ?R_1, z^P, a^P)) = (\lambda z . (\lambda! . ?R_1, z^P, b^P))]$ in *v*
 by (rule $\exists E$)
 have $[(\lambda! . ?R_1, a^P, a^P)]$ in *v*
 apply (rule *beta-C-meta-2[equiv-rl]*)
 apply *show-proper*
 using *oth-class-taut-4-a[THEN $\forall I$]* by *fast*
 hence $[(\lambda \ z . (\lambda! . ?R_1, z^P, a^P), a^P)]$ in *v*
 apply – apply (rule *beta-C-meta-1[equiv-rl]*)
 apply *show-proper*
 by *auto*
 hence $[(\lambda \ z . (\lambda! . ?R_1, z^P, b^P), a^P)]$ in *v*
 using *ab-prop[conj2] l-identity[axiom-instance, deduction, deduction]*
 by *fast*
 hence $[(\lambda! . ?R_1, a^P, b^P)]$ in *v*
 apply (*safe intro!*: *beta-C-meta-1* [where $\varphi =$

```

      λz . (λ2 (λx y. ∀ F. (F, xP) ≡ (F, yP)), z, bP), equiv-lr]
    by show-proper
  moreover have IsProperInXY (λx y. ∀ F. (F, x) ≡ (F, y))
    by show-proper
  ultimately have [∀ F. (F, aP) ≡ (F, bP) in v]
    using beta-C-meta-2[equiv-lr] by blast
  hence [(A!, aP) & (A!, bP) & a ≠ b & (∀ F. (F, aP) ≡ (F, bP)) in v]
    using ab-prop[conjI] & I by presburger
  hence [∃ y . (A!, aP) & (A!, yP) & a ≠ y & (∀ F. (F, aP) ≡ (F, yP)) in v]
    using ∃ I by fast
  thus ?thesis using ∃ I by fast
qed

```

9.13 Propositional Properties

lemma *prop-prop2-1*:

```

[∀ p . ∃ F . F = (λ x . p) in v]
proof (rule ∀ I)
  fix p
  have [(λ x . p) = (λ x . p) in v]
    using id-eq-prop-prop-1 by auto
  thus [∃ F . F = (λ x . p) in v]
    by PLM-solver
qed

```

lemma *prop-prop2-2*:

```

[F = (λ x . p) → □(∀ x . (F, xP) ≡ p) in v]
proof (rule CP)
  assume 1: [F = (λ x . p) in v]
  {
    fix v
    {
      fix x
      have [(λ x . p), xP] ≡ p in v]
        apply (rule beta-C-meta-1)
        by show-proper
    }
    hence [∀ x . [(λ x . p), xP] ≡ p in v]
      by (rule ∀ I)
  }
  hence [□(∀ x . [(λ x . p), xP] ≡ p) in v]
    by (rule RN)
  thus [□(∀ x . (F, xP) ≡ p) in v]
    using l-identity[axiom-instance, deduction, deduction,
      OF 1 [THEN id-eq-prop-prop-2 [deduction]]] by fast
qed

```

lemma *prop-prop2-3*:

```

[Propositional F → □(Propositional F) in v]
proof (rule CP)
  assume [Propositional F in v]
  hence [∃ p . F = (λ x . p) in v]
    unfolding Propositional-def .
  then obtain q where [F = (λ x . q) in v]
    by (rule ∃ E)
  hence [□(F = (λ x . q)) in v]
    using id-nec[equiv-lr] by auto
  hence [∃ p . □(F = (λ x . p)) in v]
    using ∃ I by fast
  thus [□(Propositional F) in v]
    unfolding Propositional-def
    using sign-S5-thm-1 [deduction] by fast
qed

```

lemma *prop-indis*:
 $[Indiscriminate\ F \rightarrow (\neg(\exists\ x\ y.\ (F, x^P) \ \&\ (\neg(F, y^P))))\ in\ v]$
proof (*rule CP*)
 assume $[Indiscriminate\ F\ in\ v]$
 hence 1: $[\Box((\exists\ x.\ (F, x^P)) \rightarrow (\forall\ x.\ (F, x^P)))\ in\ v]$
 unfolding *Indiscriminate-def* .
 {
 assume $[\exists\ x\ y.\ (F, x^P) \ \&\ \neg(F, y^P)\ in\ v]$
 then obtain x where $[\exists\ y.\ (F, x^P) \ \&\ \neg(F, y^P)\ in\ v]$
 by (*rule* $\exists E$)
 then obtain y where 2: $[(F, x^P) \ \&\ \neg(F, y^P)\ in\ v]$
 by (*rule* $\exists E$)
 hence $[\exists\ x.\ (F, x^P)\ in\ v]$
 using $\&E(1)\ \exists I$ by *fast*
 hence $[\forall\ x.\ (F, x^P)\ in\ v]$
 using 1[*THEN qml-2[axiom-instance, deduction], deduction*] by *fast*
 hence $[(F, y^P)\ in\ v]$
 using *cqt-orig-1[deduction]* by *fast*
 hence $[(F, y^P) \ \&\ (\neg(F, y^P))\ in\ v]$
 using 2 $\&I$ $\&E$ by *fast*
 hence $[\neg(\exists\ x\ y.\ (F, x^P) \ \&\ \neg(F, y^P))\ in\ v]$
 using *pl-1[axiom-instance, deduction, THEN modus-tollens-1]*
 oth-class-taut-1-a by *blast*
 }
 thus $[\neg(\exists\ x\ y.\ (F, x^P) \ \&\ \neg(F, y^P))\ in\ v]$
 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 $[\Box(Propositional\ F)\ in\ v]$
 using *prop-prop2-3[deduction]* by *auto*
 moreover {
 fix w
 assume $[\exists\ p.\ (F = (\lambda\ y.\ p))\ in\ w]$
 then obtain q where *q-prop*: $[F = (\lambda\ y.\ q)\ in\ w]$
 by (*rule* $\exists E$)
 {
 assume $[\exists\ x.\ (F, x^P)\ in\ w]$
 then obtain a where $[(F, a^P)\ in\ w]$
 by (*rule* $\exists E$)
 hence $[(\lambda\ y.\ q, a^P)\ in\ w]$
 using *q-prop l-identity[axiom-instance, deduction, deduction]* by *fast*
 hence $q: [q\ in\ w]$
 apply (*safe intro!*: *beta-C-meta-1[where $\varphi = \lambda y. q$, equiv-lr]*)
 apply *show-proper*
 by *simp*
 {
 fix x
 have $[(\lambda\ y.\ q, x^P)\ in\ w]$
 apply (*safe intro!*: *q beta-C-meta-1[equiv-rl]*)
 by *show-proper*
 hence $[(F, x^P)\ in\ w]$
 using *q-prop[eq-sym] l-identity[axiom-instance, deduction, deduction]*
 by *fast*
 }
 }
 hence $[\forall\ x.\ (F, x^P)\ in\ w]$
 by (*rule* $\forall I$)
 }
 thus $[\Box(Propositional\ F)\ in\ v]$
 using *prop-prop2-3[deduction]* by *auto*
qed

```

    }
    hence  $[(\exists x . \langle F, x^P \rangle) \rightarrow (\forall x . \langle F, x^P \rangle)]$  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  $[(\neg(\exists x . \langle F, x^P \rangle)) \rightarrow ((\exists x . \langle F, x^P \rangle) \rightarrow (\forall x . \langle F, x^P \rangle))]$  in  $w$ 
    using useful-tautologies-3 by auto
    hence  $[(\forall x . \neg\langle F, x^P \rangle) \rightarrow ((\exists x . \langle F, x^P \rangle) \rightarrow (\forall x . \langle F, x^P \rangle))]$  in  $w$ 
    apply - apply (PLM-subst-method  $\neg(\exists x . \langle F, x^P \rangle) (\forall x . \neg\langle F, x^P \rangle)$ )
    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:
   $[\neg(Indiscriminate\ (E!))\ in\ v]$ 
  proof (rule reductio-aa-2)
    show  $[\Box\neg(\forall x . \langle E!, x^P \rangle)]$  in  $v$ 
    using a-objects-exist-3 .
  next
    assume  $[Indiscriminate\ E!\ in\ v]$ 
    thus  $[\neg\Box\neg(\forall x . \langle E!, x^P \rangle)]$  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:
   $[\neg(Indiscriminate\ (E!^-))\ in\ v]$ 
  proof (rule reductio-aa-2)
    assume  $[Indiscriminate\ (E!^-)\ in\ v]$ 
    moreover have  $[\Box(\exists x . \langle E!^-, x^P \rangle)]$  in  $v$ 
    apply (PLM-subst-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-subst-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 -
  apply (PLM-subst-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 -
    apply (PLM-subst-method  $\neg\neg(\forall x. \neg(O!, x^P)) \forall x. \neg(O!, x^P)$ )
    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 [ $\neg(\forall x. (O!, x^P))$  in v]
    using a-objects-exist-2[THEN qml-2[axiom-instance, deduction]]
    by blast
next
  assume [Indiscriminate O! in v]
  thus [ $(\forall x. (O!, x^P))$  in v]
    unfolding Indiscriminate-def
    using o-objects-exist-2 qml-1[axiom-instance, deduction, deduction]
    qml-2[axiom-instance, deduction] by blast
qed

lemma prop-in-f-3-d:
  [ $\neg(\text{Indiscriminate } (A!))$  in v]
proof (rule reductio-aa-2)
  show [ $\neg(\forall x. (A!, x^P))$  in v]
    using o-objects-exist-3[THEN qml-2[axiom-instance, deduction]]
    by blast
next
  assume [Indiscriminate A! in v]
  thus [ $(\forall x. (A!, x^P))$  in v]
    unfolding Indiscriminate-def
    using a-objects-exist-1 qml-1[axiom-instance, deduction, deduction]
    qml-2[axiom-instance, deduction] by blast
qed

lemma prop-in-f-4-a:
  [ $\neg(\text{Propositional } E!)$  in v]
  using prop-in-thm[deduction] prop-in-f-3-a modus-tollens-1 CP
  by meson

lemma prop-in-f-4-b:
  [ $\neg(\text{Propositional } (E!^-))$  in v]
  using prop-in-thm[deduction] prop-in-f-3-b modus-tollens-1 CP
  by meson

lemma prop-in-f-4-c:
  [ $\neg(\text{Propositional } (O!))$  in v]
  using prop-in-thm[deduction] prop-in-f-3-c modus-tollens-1 CP
  by meson

lemma prop-in-f-4-d:
  [ $\neg(\text{Propositional } (A!))$  in v]
  using prop-in-thm[deduction] prop-in-f-3-d modus-tollens-1 CP
  by meson

lemma prop-prop-nec-1:
  [ $\Diamond(\exists p. F = (\lambda x. p)) \rightarrow (\exists p. F = (\lambda x. p))$  in v]
proof (rule CP)
  assume [ $\Diamond(\exists p. F = (\lambda x. p))$  in v]
  hence [ $\exists p. \Diamond(F = (\lambda x. p))$  in v]
    using BF $\Diamond$ [deduction] by auto
  then obtain p where [ $\Diamond(F = (\lambda x. p))$  in v]
    by (rule  $\exists E$ )
  hence [ $\Diamond(\forall x. \llbracket x^P, F \rrbracket \equiv \llbracket x^P, \lambda x. p \rrbracket)$  in v]

```

```

    unfolding identity-defs .
  hence  $[\Box(\forall x. \llbracket x^P, F \rrbracket \equiv \llbracket x^P, \lambda x. p \rrbracket) \text{ in } v]$ 
    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 Buridan $\Diamond$ [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 fast
      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 - 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 $x \equiv \langle A!, x \rangle \ \& \ (\forall F. \langle x, F \rangle \rightarrow \text{Propositional } F)$

definition *EncodeProposition* (infixl Σ 70) **where**

$x \Sigma p \equiv \langle A!, x \rangle \ \& \ \langle x, \lambda x. p \rangle$

definition *TrueInSituation* (infixl \models 10) **where**

$x \models p \equiv \text{Situation } x \ \& \ x \Sigma p$

definition *PossibleWorld* **where**

PossibleWorld $x \equiv \text{Situation } x \ \& \ \Diamond(\forall p. x \Sigma p \equiv p)$

10.2 Auxiliary Lemmas

lemma *possit-sit-1*:

$[\text{Situation } (x^P) \equiv \Box(\text{Situation } (x^P)) \text{ in } v]$

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

assume $[\text{Situation } (x^P) \text{ in } v]$

hence 1: $[\langle A!, x^P \rangle \ \& \ (\forall F. \langle x^P, F \rangle \rightarrow \text{Propositional } F) \text{ in } v]$

unfolding *Situation-def* **by** *auto*

have $[\Box \langle A!, x^P \rangle \text{ in } v]$

using 1[*conj1*, *THEN oa-facts-2[deduction]*].

moreover have $[\Box(\forall F. \langle x^P, F \rangle \rightarrow \text{Propositional } F) \text{ in } v]$

using 1[*conj2*] **unfolding** *Propositional-def*

by (rule *enc-prop-nec-2[deduction]*)

ultimately show $[\Box \text{Situation } (x^P) \text{ in } v]$

unfolding *Situation-def*

apply *cut-tac* **apply** (rule *KBasic-3[equiv-rl]*)

by (rule *intro-elim-1*)

next

assume $[\Box \text{Situation } (x^P) \text{ in } v]$

thus $[\text{Situation } (x^P) \text{ in } v]$

using *qml-2[axiom-instance, deduction]* **by** *auto*

qed

lemma *possworld-nec*:

$[\text{PossibleWorld } (x^P) \equiv \Box(\text{PossibleWorld } (x^P)) \text{ in } v]$

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*:

$[\Box((x^P) \models p) \equiv \Box(\Box((x^P) \models p)) \text{ in } v]$

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

assume $[x^P \models p \text{ in } v]$

hence $[\text{Situation } (x^P) \ \& \ (\langle A!, x^P \rangle \ \& \ \langle x^P, \lambda x. p \rangle) \text{ in } v]$

unfolding *TrueInSituation-def* *EncodeProposition-def*.

hence $[(\Box \text{Situation } (x^P) \ \& \ \Box \langle A!, x^P \rangle) \ \& \ \Box \langle x^P, \lambda x. p \rangle \text{ in } v]$

using **&I** **&E** *possit-sit-1[equiv-lr]* *oa-facts-2[deduction]*

encoding[axiom-instance, deduction] **by** *metis*

thus $[\Box(\Box((x^P) \models p)) \text{ in } v]$

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*:

$[(\langle A!, x^P \rangle \ \& \ (\forall F. \ (\langle x^P, F \rangle \equiv (\exists p. p \ \& \ (F = (\lambda x. p)))))) \rightarrow (\text{PossibleWorld } (x^P)) \text{ in } v]$

proof (*rule CP*)

assume *DefX*: $[(\langle A!, x^P \rangle \ \& \ (\forall F. \ (\langle x^P, F \rangle \equiv (\exists p. p \ \& \ (F = (\lambda x. p)))))) \text{ in } v]$

have [*Situation* (x^P) in v]

proof –

have $[(\langle A!, x^P \rangle \text{ in } v]$

using *DefX*[*conj1*] .

moreover have $[(\forall F. \ \langle x^P, F \rangle \rightarrow \text{Propositional } F) \text{ in } v]$

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

fix F

assume $[\langle x^P, F \rangle \text{ in } v]$

moreover have $[\langle x^P, F \rangle \equiv (\exists p. p \ \& \ (F = (\lambda x. p))) \text{ in } v]$

using *DefX*[*conj2*] *cqt-1*[*axiom-instance*, *deduction*] **by** *auto*

ultimately have $[(\exists p. p \ \& \ (F = (\lambda x. p))) \text{ in } v]$

using $\equiv E(1)$ **by** *blast*

then obtain p **where** $[p \ \& \ (F = (\lambda x. p)) \text{ in } v]$

by (*rule* $\exists E$)

hence $[(F = (\lambda x. p)) \text{ in } v]$

by (*rule* $\&E(2)$)

hence $[(\exists p. (F = (\lambda x. p))) \text{ in } v]$

by *PLM-solver*

thus [*Propositional* F in v]

unfolding *Propositional-def* .

qed

ultimately show [*Situation* (x^P) in v]

unfolding *Situation-def* **by** (*rule* $\&I$)

qed

moreover have $[\Diamond(\forall p. x^P \ \Sigma p \equiv p) \text{ in } v]$

unfolding *EncodeProposition-def*

proof (*rule* *TBasic*[*deduction*]; *rule* $\forall I$)

fix q

have *EncodeLambda*:

$[\langle x^P, \lambda x. q \rangle \equiv (\exists p. p \ \& \ ((\lambda x. q) = (\lambda x. p))) \text{ in } v]$

using *DefX*[*conj2*] **by** (*rule* *cqt-1*[*axiom-instance*, *deduction*])

moreover {

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 $[q \ \& \ ((\lambda x. q) = (\lambda x. q)) \text{ in } v]$

by (*rule* $\&I$)

hence $[\exists p. p \ \& \ ((\lambda x. q) = (\lambda x. p)) \text{ in } v]$

by *PLM-solver*

moreover have $[(\langle A!, x^P \rangle \text{ in } v]$

using *DefX*[*conj1*] .

ultimately have $[(\langle A!, x^P \rangle \ \& \ \langle x^P, \lambda x. q \rangle \text{ in } v]$

using *EncodeLambda*[*equiv-rl*] $\&I$ **by** *auto*

}

moreover {

assume $[(\langle A!, x^P \rangle \ \& \ \langle x^P, \lambda x. q \rangle \text{ in } v]$

hence $[\langle x^P, \lambda x. q \rangle \text{ in } v]$

using $\&E(2)$ **by** *auto*

hence $[\exists p. p \ \& \ ((\lambda x. q) = (\lambda x. p)) \text{ in } v]$

using *EncodeLambda*[*equiv-lr*] **by** *auto*

then obtain p **where** *p-and-lambda-q-is-lambda-p*:

$[p \ \& \ ((\lambda x. q) = (\lambda x. p)) \text{ in } v]$

by (*rule* $\exists E$)

```

    have  $[(\lambda x . p), x^P] \equiv p$  in  $v$ 
    apply (rule beta-C-meta-1)
    by show-proper
  hence  $[(\lambda x . p), x^P]$  in  $v$ 
    using p-and-lambda-q-is-lambda-p[conj1]  $\equiv E(2)$  by auto
  hence  $[(\lambda x . q), x^P]$  in  $v$ 
    using p-and-lambda-q-is-lambda-p[conj2, THEN id-eq-prop-prop-2[deduction]]
    l-identity[axiom-instance, deduction, deduction] by fast
  moreover have  $[(\lambda x . q), x^P] \equiv q$  in  $v$ 
    apply (rule beta-C-meta-1) by show-proper
  ultimately have  $[q]$  in  $v$ 
    using  $\equiv E(1)$  by blast
}
ultimately show  $[(\lambda x . q), x^P] \equiv q$  in  $v$ 
  using &I  $\equiv I$  CP by auto
qed

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

```

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

theorem *SemanticPossibleWorldForSyntacticPossibleWorlds:*

$\forall x . [PossibleWorld (x^P)] \longrightarrow$
 $(\exists v . \forall p . [(x^P \models p)] \longleftrightarrow [p \text{ in } v])$

proof

```

fix x
{
  assume PossWorldX:  $[PossibleWorld (x^P)]$  in  $w$ 
  hence SituationX:  $[Situation (x^P)]$  in  $w$ 
    unfolding PossibleWorld-def apply cut-tac by PLM-solver
  have PossWorldExpanded:
     $[(\lambda x . x^P) \& (\forall F . \llbracket x^P, F \rrbracket \rightarrow (\exists p . F = (\lambda x . p)))]$ 
     $\& \Diamond(\forall p . \llbracket A!, x^P \rrbracket \& \llbracket x^P, \lambda x . p \rrbracket \equiv p)$  in  $w$ 
    using PossWorldX
    unfolding PossibleWorld-def Situation-def
    Propositional-def EncodeProposition-def .
  have AbstractX:  $[(\lambda x . x^P)]$  in  $w$ 
    using PossWorldExpanded[conj1, conj1] .

  have  $[\Diamond(\forall p . \llbracket x^P, \lambda x . p \rrbracket \equiv p)]$  in  $w$ 
    apply (PLM-subst-method
       $\lambda p . \llbracket A!, x^P \rrbracket \& \llbracket x^P, \lambda x . p \rrbracket$ 
       $\lambda p . \llbracket x^P, \lambda x . p \rrbracket$ )
    subgoal using PossWorldExpanded[conj1, conj1, THEN oa-facts-2[deduction]]
      using Semantics.T6 apply cut-tac by PLM-solver
    using PossWorldExpanded[conj2] .

  hence  $\exists v . \forall p . (\llbracket x^P, \lambda x . p \rrbracket \text{ in } v)$ 
    =  $[p \text{ in } v]$ 
    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:*

```

   $\forall p . (\llbracket x^P, \lambda x . p \rrbracket \text{ in } v) = [p \text{ in } v]$ 
  by auto
{
  fix p
  {

```

```

    assume  $[(x^P) \models p \text{ in } w]$ 
    hence  $[(x^P) \models p \text{ in } v]$ 
      using TrueInWorldNecc[equiv-lr] Semantics.T6 by simp
    hence  $[ \textit{Situation } (x^P) \ \& \ (\!|A!, x^P|) \ \& \ |x^P, \lambda x. p| \text{ in } v ]$ 
      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 TrueInWorldNecc[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  $[ \textit{PossibleWorld } (x^P) \text{ in } w ] \longrightarrow$ 
   $(\exists v . \forall p . [(x^P) \models p \text{ in } w] \longleftrightarrow [p \text{ in } v])$ 
  by blast
qed

```

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

theorem *SyntacticPossibleWorldForSemanticPossibleWorlds:*

$\forall v . \exists x . [\textit{PossibleWorld } (x^P) \text{ in } w] \wedge$
 $(\forall p . [p \text{ in } v] \longleftrightarrow [((x^P) \models p) \text{ in } w])$

proof

fix v

have $[\exists x . (\!|A!, x^P|) \ \& \ (\forall F . (|x^P, F| \equiv$
 $(\exists p . p \ \& \ (F = (\lambda x . p)))) \text{ in } v]$

using *A-objects*[*axiom-instance*] by *fast*

then obtain x where *DefX*:

$[(\!|A!, x^P|) \ \& \ (\forall F . (|x^P, F| \equiv (\exists p . p \ \& \ (F = (\lambda x . p)))) \text{ in } v]$
 by (rule $\exists E$)

hence *PossWorldX*: $[\textit{PossibleWorld } (x^P) \text{ in } v]$

using *PossWorldAux*[*deduction*] by *blast*

hence $[\textit{PossibleWorld } (x^P) \text{ in } w]$

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-rl]
  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:
     $[(p \ \& \ ((\lambda \ x \ . \ q) = (\lambda \ x \ . \ p))) \text{ in } v]$ 
    by (rule  $\exists E$ )
  have  $[\llbracket (\lambda \ x \ . \ p), x^P \rrbracket \equiv p \text{ in } v]$ 
    apply (rule beta-C-meta-1)
    by show-proper
  hence  $[\llbracket (\lambda \ x \ . \ q), x^P \rrbracket \equiv p \text{ in } v]$ 
    using l-identity[where  $\beta = (\lambda \ x \ . \ q)$  and  $\alpha = (\lambda \ x \ . \ p)$ ,
      axiom-instance, deduction, deduction]
    using 4[conj2, THEN id-eq-prop-prop-2[deduction]] by meson
  hence  $[\llbracket (\lambda \ x \ . \ q), x^P \rrbracket \text{ in } v]$  using 4[conj1]  $\equiv E(2)$  by blast
  moreover have  $[\llbracket (\lambda \ x \ . \ q), x^P \rrbracket \equiv q \text{ in } v]$ 
    apply (rule beta-C-meta-1)
    by show-proper
  ultimately have  $[q \text{ in } v]$ 
    using  $\equiv E(1)$  by blast
}
ultimately show  $[q \text{ in } v] \longleftrightarrow [(x^P) \models q \text{ in } w]$ 
  by blast
qed
ultimately show  $\exists \ x \ . \ [PossibleWorld \ (x^P) \text{ in } w]$ 
   $\wedge (\forall \ p \ . \ [p \text{ in } v] \longleftrightarrow [(x^P) \models p \text{ in } w])$ 
  by auto
qed
end

```

11 Artificial Theorems

Remark 17. *Some examples of theorems that can be derived from the model structure, but which are not derivable from the deductive system PLM itself.*

locale *ArtificialTheorems*
begin

lemma *lambda-enc-1*:
 $[(\lambda x . \llbracket x^P, F \rrbracket \equiv \llbracket x^P, F \rrbracket, y^P) \text{ in } v]$
by (*auto simp: meta-defs meta-aux conn-defs forall- Π_1 -def*)

lemma *lambda-enc-2*:
 $[(\lambda x . \llbracket y^P, G \rrbracket, x^P) \equiv \llbracket y^P, G \rrbracket \text{ in } v]$
by (*auto simp: meta-defs meta-aux conn-defs forall- Π_1 -def*)

Remark 18. *The following is not a theorem and nitpick can find a countermodel. This is expected and important. If this were a theorem, the theory would become inconsistent.*

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

Remark 19. *Instead the following two statements hold.*

lemma *lambda-enc-4*:
 $[(\lambda x . \llbracket x^P, F \rrbracket, x^P) \text{ in } v] = (\exists y . \nu\nu y = \nu\nu x \wedge [\llbracket y^P, F \rrbracket \text{ in } v])$
by (*simp add: meta-defs meta-aux*)

lemma *lambda-ex*:
 $[(\lambda x . \varphi(x^P), x^P) \text{ in } v] = (\exists y . \nu\nu y = \nu\nu x \wedge [\varphi(y^P) \text{ in } v])$
by (*simp add: meta-defs meta-aux*)

Remark 20. *These statements can be translated to statements in the embedded logic.*

lemma *lambda-ex-emb*:
 $[(\lambda x . \varphi(x^P), x^P) \equiv (\exists y . (\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P)) \text{ in } v]$
proof(*rule MetaSolver.EquivI*)
interpret *MetaSolver* .
{
assume $[(\lambda x . \varphi(x^P), x^P) \text{ in } v]$
then obtain *y* **where** $\nu\nu y = \nu\nu x \wedge [\varphi(y^P) \text{ in } v]$
using *lambda-ex* **by** *blast*
moreover hence $[(\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \text{ in } v]$
apply — **apply** *meta-solver*
by (*simp add: Semantics.d_κ-proper Semantics.ex1-def*)
ultimately have $[\exists y . (\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P) \text{ in } v]$
using *ExIRule ConjI* **by** *fast*
}
moreover {
assume $[\exists y . (\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P) \text{ in } v]$
then obtain *y* **where** *y-def*: $[(\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P) \text{ in } v]$
by (*rule ExERule*)
hence $\bigwedge F . [\llbracket F, x^P \rrbracket \text{ in } v] = [\llbracket F, y^P \rrbracket \text{ in } v]$
apply — **apply** (*drule ConjE*) **apply** (*drule conjunct1*)
apply (*drule AllE*) **apply** (*drule EquivE*) **by** *simp*
hence $[(\text{make}\Pi_1 (\lambda u s w . \nu\nu y = u), x^P) \text{ in } v]$
 $= [(\text{make}\Pi_1 (\lambda u s w . \nu\nu y = u), y^P) \text{ in } v]$ **by** *auto*
hence $\nu\nu y = \nu\nu x$ **by** (*simp add: meta-defs meta-aux*)
moreover have $[\varphi(y^P) \text{ in } v]$ **using** *y-def ConjE* **by** *blast*
ultimately have $[(\lambda x . \varphi(x^P), x^P) \text{ in } v]$
using *lambda-ex* **by** *blast*
}
ultimately show $[(\lambda x . \varphi(x^P), x^P) \text{ in } v]$
 $= [\exists y . (\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P) \text{ in } v]$
by *auto*
qed

lemma *lambda-enc-emb*:

$[(\lambda x . \llbracket x^P, F \rrbracket), x^P] \equiv (\exists y . (\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \llbracket y^P, F \rrbracket) \text{ in } v]$
using *lambda-ex-emb* **by** *fast*

Remark 21. *In the case of proper maps, the generalized β -conversion reduces to classical β -conversion.*

lemma *proper-beta*:
assumes *IsProperInX* φ
shows $[(\exists y . (\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P)) \equiv \varphi(x^P) \text{ in } v]$
proof (*rule MetaSolver.EquivI*; *rule*)
interpret *MetaSolver* .
assume $[\exists y . (\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P) \text{ in } v]$
then obtain y **where** $y\text{-def: } [(\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P) \text{ in } v]$ **by** (*rule ExERule*)
hence $[(\text{make}\Pi_1 (\lambda u s w . \nu\nu y = u), x^P) \text{ in } v] = [(\text{make}\Pi_1 (\lambda u s w . \nu\nu y = u), y^P) \text{ in } v]$
using *EquivS Alle ConjE* **by** *blast*
hence $\nu\nu y = \nu\nu x$ **by** (*simp add: meta-defs meta-aux*)
thus $[\varphi(x^P) \text{ in } v]$
using $y\text{-def}[THEN ConjE[THEN conjunct2]]$
assms IsProperInX.rep-eq valid-in.rep-eq
by *blast*
next
interpret *MetaSolver* .
assume $[\varphi(x^P) \text{ in } v]$
moreover have $[\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, x^P \rrbracket \text{ in } v]$ **apply** *meta-solver* **by** *blast*
ultimately show $[\exists y . (\forall F . \llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \ \& \ \varphi(y^P) \text{ in } v]$
by (*meson ConjI ExI*)
qed

Remark 22. *The following theorem is a consequence of the constructed Aczel-model, but not part of PLM. Separate research on possible modifications of the embedding suggest that this artificial theorem can be avoided by introducing a dependency on states for the mapping from abstract objects to special urelements.*

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

end

12 Sanity Tests

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

12.1 Consistency

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

12.2 Intensionality

```

lemma [( $\lambda y. (q \vee \neg q)$ ) = ( $\lambda y. (p \vee \neg p)$ ) in v]
  unfolding identity- $\Pi_1$ -def conn-defs
  apply (rule Eq1I) apply (simp add: meta-defs)
  nitpick[expect = genuine, user-axioms=true, card i = 2,
    card j = 2, card  $\omega$  = 1, card  $\sigma$  = 1,
    sat-solver = MiniSat-JNI, verbose, show-all]
  oops — Countermodel by Nitpick
lemma [( $\lambda y. (p \vee q)$ ) = ( $\lambda y. (q \vee p)$ ) in v]
  unfolding identity- $\Pi_1$ -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  $\sigma$  = 1, card  $\omega$  = 1,
    card v = 2, verbose, show-all]
  oops — Countermodel by Nitpick

```

12.3 Concreteness coindices with Object Domains

```

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

```

12.4 Justification for Meta-Logical Axioms

Remark 23. *OrdinaryObjectsPossiblyConcreteAxiom is equivalent to "all ordinary objects are possibly concrete".*

```

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

```

Remark 24. *OrdinaryObjectsPossiblyConcreteAxiom is equivalent to "all abstract objects are necessarily not concrete".*

```

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

```

Remark 25. *PossiblyContingentObjectExistsAxiom is equivalent to the corresponding statement in the embedded logic.*

```

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

```

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

```

lemma PossiblyNoContingentObjectExistsCheck:
  PossiblyNoContingentObjectExists  $\longleftrightarrow$   $[\neg(\Box(\neg(\forall x. \langle E!, x^P \rangle \rightarrow \Box(\langle E!, x^P \rangle)))))$  in  $v$ ]
apply (simp add: meta-defs forall- $\nu$ -def meta-aux split:  $\nu$ .split  $v$ .split)
using  $\nu\nu$ - $\omega\nu$ -is- $\omega\nu$  by blast

```

12.5 Relations in the Meta-Logic

Remark 27. *Material equality in the embedded logic corresponds to equality in the actual state in the meta-logic.*

```

lemma mat-eq-is-eq-dj:
   $[\forall x. \Box(\langle F, x^P \rangle \equiv \langle G, x^P \rangle)$  in  $v$ ]  $\longleftrightarrow$ 
   $((\lambda x. (eval\Pi_1 F) x dj) = (\lambda x. (eval\Pi_1 G) x dj))$ 
proof
  assume 1:  $[\forall x. \Box(\langle F, x^P \rangle \equiv \langle G, x^P \rangle)$  in  $v$ ]
  {
    fix  $v$ 
    fix  $y$ 
    obtain  $x$  where  $y$ -def:  $y = \nu\nu x$ 
    by (meson  $\nu\nu$ -surj surj-def)
    have  $(\exists r o_1. \text{Some } r = d_1 F \wedge \text{Some } o_1 = d_\kappa(x^P) \wedge o_1 \in ex1 r v) =$ 
       $(\exists r o_1. \text{Some } r = d_1 G \wedge \text{Some } o_1 = d_\kappa(x^P) \wedge o_1 \in ex1 r v)$ 
    using 1 apply – by meta-solver
    moreover obtain  $r$  where  $r$ -def:  $\text{Some } r = d_1 F$ 
    unfolding  $d_1$ -def by auto
    moreover obtain  $s$  where  $s$ -def:  $\text{Some } s = d_1 G$ 
    unfolding  $d_1$ -def by auto
    moreover have  $\text{Some } x = d_\kappa(x^P)$ 
    using  $d_\kappa$ -proper by simp
    ultimately have  $(x \in ex1 r v) = (x \in ex1 s v)$ 
    by (metis option.inject)
    hence  $(eval\Pi_1 F) y dj v = (eval\Pi_1 G) y dj v$ 
    using  $r$ -def  $s$ -def  $y$ -def by (simp add:  $d_1$ .rep-eq  $ex1$ -def)
  }
  thus  $(\lambda x. eval\Pi_1 F x dj) = (\lambda x. eval\Pi_1 G x dj)$ 
  by auto
next
  assume 1:  $(\lambda x. eval\Pi_1 F x dj) = (\lambda x. eval\Pi_1 G x dj)$ 
  {
    fix  $y v$ 
    obtain  $x$  where  $x$ -def:  $x = \nu\nu y$ 
    by simp
    hence  $eval\Pi_1 F x dj = eval\Pi_1 G x dj$ 
    using 1 by metis
    moreover obtain  $r$  where  $r$ -def:  $\text{Some } r = d_1 F$ 
    unfolding  $d_1$ -def by auto
    moreover obtain  $s$  where  $s$ -def:  $\text{Some } s = d_1 G$ 
    unfolding  $d_1$ -def by auto
    ultimately have  $(y \in ex1 r v) = (y \in ex1 s v)$ 
    by (simp add:  $d_1$ .rep-eq  $ex1$ -def  $\nu\nu$ -surj  $x$ -def)
    hence  $\langle F, y^P \rangle \equiv \langle G, y^P \rangle$  in  $v$ ]
    apply – apply meta-solver
    using  $r$ -def  $s$ -def by (metis Semantics. $d_\kappa$ -proper option.inject)
  }
  thus  $[\forall x. \Box(\langle F, x^P \rangle \equiv \langle G, x^P \rangle)$  in  $v$ ]
  using T6 T8 by fast
qed

```

Remark 28. *Materially equivalent relations are equal in the embedded logic if and only if they also coincide in all other states.*


```

lemma mat-eq-is-eq-if-eq-forall-j:
  assumes  $[\forall x. \Box(\llbracket F, x^P \rrbracket \equiv \llbracket G, x^P \rrbracket)]$  in v
  shows  $[F = G \text{ in } v] \longleftrightarrow$ 
     $(\forall s. s \neq dj \longrightarrow (\forall x. (eval\Pi_1 F) x s = (eval\Pi_1 G) x s))$ 
proof
  interpret MetaSolver .
  assume  $[F = G \text{ in } v]$ 
  hence  $F = G$ 
    apply – unfolding identity- $\Pi_1$ -def by meta-solver
  thus  $\forall s. s \neq dj \longrightarrow (\forall x. eval\Pi_1 F x s = eval\Pi_1 G x s)$ 
    by auto
next
  interpret MetaSolver .
  assume  $\forall s. s \neq dj \longrightarrow (\forall x. eval\Pi_1 F x s = eval\Pi_1 G x s)$ 
  moreover have  $((\lambda x. (eval\Pi_1 F) x dj) = (\lambda x. (eval\Pi_1 G) x dj))$ 
    using assms mat-eq-is-eq-dj by auto
  ultimately have  $\forall s x. eval\Pi_1 F x s = eval\Pi_1 G x s$ 
    by metis
  hence  $eval\Pi_1 F = eval\Pi_1 G$ 
    by blast
  hence  $F = G$ 
    by  $(metis\ eval\Pi_1\text{-inverse})$ 
  thus  $[F = G \text{ in } v]$ 
    unfolding identity- $\Pi_1$ -def using Eq1I by auto
qed

```

Remark 29. *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  $[\forall x. \Box(\llbracket F, x^P \rrbracket \equiv \llbracket G, x^P \rrbracket)]$  in v  $\longleftrightarrow [F = G \text{ in } v]$ 
by  $(metis\ (no\text{-types})\ MetaSolver.Eq_1S\ assms\ identity\text{-}\Pi_1\text{-def}\ mat\text{-eq-is-eq-dj}\ mat\text{-eq-is-eq-if-eq-forall-j})$ 

```

12.6 Lambda Expressions

```

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

```

```

lemma lambda-interpret-2:
assumes  $[a = (\iota y. \llbracket G, y^P \rrbracket)]$  in v
shows  $(\lambda x. \llbracket R, x^P, a \rrbracket) = (\lambda x. \llbracket R, x^P, \iota y. \llbracket G, y^P \rrbracket \rrbracket)$ 
proof –
  have  $a = (\iota y. \llbracket G, y^P \rrbracket)$ 
    using MetaSolver.Eq $\kappa$ S Semantics.d $\kappa$ -inject assms
      identity- $\kappa$ -def by auto
  thus ?thesis by simp
qed

```

end

```

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

```

13 Paradox

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

13.1 Auxiliary Lemmas

lemma *exe-impl-exists*:

```

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

```

lemma *exists-unique-actual-equiv*:

```

[[ $(\exists !y. \mathcal{A}(y = x \ \& \ \psi \ (x^P)))$ ]]  $\equiv \mathcal{A}\psi \ (x^P)$  in  $v$ 
proof (rule  $\equiv I$ ; rule  $CP$ )
  fix  $x \ v$ 
  let  $?\varphi = \lambda y \ x. y = x \ \& \ \psi \ (x^P)$ 
  assume  $[\exists !y. \mathcal{A}?\varphi \ y \ x]$  in  $v$ 
  hence  $[\exists \alpha. \mathcal{A}?\varphi \ \alpha \ x \ \& \ (\forall \beta. \mathcal{A}?\varphi \ \beta \ x \rightarrow \beta = \alpha)]$  in  $v$ 
    unfolding exists-unique-def by simp
  then obtain  $\alpha$  where
    [ $\mathcal{A}?\varphi \ \alpha \ x \ \& \ (\forall \beta. \mathcal{A}?\varphi \ \beta \ x \rightarrow \beta = \alpha)$  in  $v$ ]
    by (rule Instantiate)
  hence [ $\mathcal{A}(\alpha = x \ \& \ \psi \ (x^P))$  in  $v$ ]
    using  $\&E$  by blast
  thus [ $\mathcal{A}(\psi \ (x^P))$  in  $v$ ]
    using Act-Basic-2[equiv-lr]  $\&E$  by blast

```

```

next
  fix x v
  let ?φ = λ y x. y = x & ψ (xP)
  assume 1: [Aψ (xP) in v]
  have [x = x in v]
    using id-eq-1 [where 'a=ν] by simp
  hence [A(x = x) in v]
    using id-act-3 [equiv-lr] by fast
  hence [A(x = x & ψ (xP)) in v]
    using 1 Act-Basic-2 [equiv-rl] &I by blast
  hence [A?φ x x in v]
    by simp
  moreover have [∀ β. A?φ β x → β = x in v]
  proof (rule ∀ I; rule CP)
    fix β
    assume [A?φ β x in v]
    hence [A(β = x) in v]
      using Act-Basic-2 [equiv-lr] &E by fast
    thus [β = x in v] using id-act-3 [equiv-rl] by fast
  qed
  ultimately have [A?φ x x & (∀ β. A?φ β x → β = x) in v]
    using &I by fast
  hence [∃ α. A?φ α x & (∀ β. A?φ β x → β = α) in v]
    by (rule existential)
  thus [∃ !y. A?φ y x in v]
    unfolding exists-unique-def by simp
qed

```

13.2 Fake β-Conversion using Description Backdoor

definition *backdoor* where

backdoor ≡ λ ψ . λ x . (λ x . ∀ p . p → p), λ y . y = x & ψ (x^P))

lemma *fake-beta*:

```

assumes ∧ G φ. IsProperInX (λ x . (λ G, λ y . φ y x))
shows [(λ backdoor (λ x . ψ x), xP) ≡ Aψ (xP) in v]
proof (rule ≡I; rule CP)
  assume [(λ backdoor ψ, xP) in v]
  hence [(λ x. ∀ p. p → p, λ y. y = x & ψ (xP)) in v]
    using beta-C-meta-1 [equiv-lr, OF assms]
    unfolding backdoor-def identity-ν-def by fast
  hence [∃ !y. A (y = x & ψ (xP)) in v]
    using exe-impl-exists [equiv-rl] by fast
  thus [Aψ (xP) in v]
    using exists-unique-actual-equiv [equiv-lr] by blast
next
  assume [Aψ (xP) in v]
  hence [∃ !y. A (y = x & ψ (xP)) in v]
    using exists-unique-actual-equiv [equiv-rl] by blast
  hence [(λ x. ∀ p. p → p, λ y. y = x & ψ (xP)) in v]
    using exe-impl-exists [equiv-rl] by fast
  thus [(λ backdoor ψ, xP) in v]
    using beta-C-meta-1 [equiv-rl, OF assms]
    unfolding backdoor-def unfolding identity-ν-def by fast
qed

```

lemma *fake-beta-act*:

```

assumes ∧ G φ. IsProperInX (λ x . (λ G, λ y . φ y x))
shows [(λ backdoor (λ x . ψ x), xP) ≡ ψ (xP) in dw]
using fake-beta [OF assms]
  logic-actual [necessitation-averse-axiom-instance]
  intro-elim-6-e by blast

```

13.3 Resulting Paradox

```

lemma paradox:
  assumes  $\bigwedge G \varphi. \text{IsProperInX } (\lambda x. \langle G, \iota y. \varphi y x \rangle)$ 
  shows False
proof –
  obtain K where K-def:
    K = backdoor  $(\lambda x. \exists F. \langle x, F \rangle \ \& \ \neg \langle F, x \rangle)$  by auto
  have  $[\exists x. \langle A!, x^P \rangle \ \& \ (\forall F. \langle x^P, F \rangle \equiv (F = K))]$  in dw
    using A-objects[axiom-instance] by fast
  then obtain x where x-prop:
     $[\langle A!, x^P \rangle \ \& \ (\forall F. \langle x^P, F \rangle \equiv (F = K))]$  in dw
    by (rule Instantiate)
  {
    assume  $[\langle K, x^P \rangle]$  in dw
    hence  $[\exists F. \langle x^P, F \rangle \ \& \ \neg \langle F, x^P \rangle]$  in dw
      unfolding K-def using fake-beta-act[OF assms, equiv-lr]
      by blast
    then obtain F where F-def:
       $[\langle x^P, F \rangle \ \& \ \neg \langle F, x^P \rangle]$  in dw by (rule Instantiate)
    hence  $[F = K]$  in dw
      using x-prop[conj2, THEN  $\forall E$ [where  $\beta = F$ ], equiv-lr]
      &E unfolding K-def by blast
    hence  $[\neg \langle K, x^P \rangle]$  in dw
      using l-identity[axiom-instance, deduction, deduction]
      F-def[conj2] by fast
  }
  hence 1:  $[\neg \langle K, x^P \rangle]$  in dw
    using reductio-aa-1 by blast
  hence  $[\neg (\exists F. \langle x^P, F \rangle \ \& \ \neg \langle F, x^P \rangle)]$  in dw
    using fake-beta-act[OF assms,
      THEN oth-class-taut-5-d[equiv-lr],
      equiv-lr]
    unfolding K-def by blast
  hence  $[\forall F. \langle x^P, F \rangle \rightarrow \langle F, x^P \rangle]$  in dw
    apply – unfolding exists-def by PLM-solver
  moreover have  $[\langle x^P, K \rangle]$  in dw
    using x-prop[conj2, THEN  $\forall E$ [where  $\beta = K$ ], equiv-rl]
    id-eq-1 by blast
  ultimately have  $[\langle K, x^P \rangle]$  in dw
    using  $\forall E$  vdash-properties-10 by blast
  hence  $\bigwedge \varphi. [\varphi]$  in dw
    using raa-cor-2 1 by blast
  thus False using Semantics.T4 by auto
qed

```

13.4 Original Version of the Paradox

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

```

lemma assumes  $\bigwedge G \varphi. \text{IsProperInX } (\lambda x. \langle G, \iota y. \varphi y x \rangle)$ 
shows Fx-equiv-xH:  $[\forall H. \exists F. \Box (\forall x. \langle F, x^P \rangle \equiv \langle x^P, H \rangle)]$  in v
proof (rule  $\forall I$ )
  fix H
  let  $?G = (\lambda x. \forall p. p \rightarrow p)$ 
  obtain  $\varphi$  where  $\varphi\text{-def}$ :  $\varphi = (\lambda y x. (y^P) = x \ \& \ \langle x, H \rangle)$  by auto
  have  $[\exists F. \Box (\forall x. \langle F, x^P \rangle \equiv \langle ?G, \iota y. \varphi y (x^P) \rangle)]$  in v
    using relations-1[OF assms] by simp
  hence 1:  $[\exists F. \Box (\forall x. \langle F, x^P \rangle \equiv (\exists !y. \mathcal{A}\varphi y (x^P)))]$  in v
    apply – apply (PLM-subst-method
       $\lambda x. \langle ?G, \iota y. \varphi y (x^P) \rangle \lambda x. (\exists !y. \mathcal{A}\varphi y (x^P))$ )
    using exe-impl-exists by auto

```

then obtain F where $F\text{-def}$: $[\Box(\forall x. \langle F, x^P \rangle \equiv (\exists !y. \mathcal{A}_\varphi y (x^P))) \text{ in } v]$
 by (rule *Instantiate*)
 moreover have 2 : $\bigwedge v x. [(\exists !y. \mathcal{A}_\varphi y (x^P)) \equiv \langle x^P, H \rangle \text{ in } v]$
 proof (rule $\equiv I$; rule *CP*)
 fix $x v$
 assume $[\exists !y. \mathcal{A}_\varphi y (x^P) \text{ in } v]$
 hence $[\exists \alpha. \mathcal{A}_\varphi \alpha (x^P) \ \& \ (\forall \beta. \mathcal{A}_\varphi \beta (x^P) \rightarrow \beta = \alpha) \text{ in } v]$
 unfolding *exists-unique-def* by *simp*
 then obtain α where $[\mathcal{A}_\varphi \alpha (x^P) \ \& \ (\forall \beta. \mathcal{A}_\varphi \beta (x^P) \rightarrow \beta = \alpha) \text{ in } v]$
 by (rule *Instantiate*)
 hence $[\mathcal{A}(\alpha^P = x^P \ \& \ \langle x^P, H \rangle) \text{ in } v]$
 unfolding $\varphi\text{-def}$ using $\&E$ by *blast*
 hence $[\mathcal{A}(\langle x^P, H \rangle) \text{ in } v]$
 using *Act-Basic-2[equiv-lr]* $\&E$ by *blast*
 thus $[\langle x^P, H \rangle \text{ in } v]$
 using *en-eq-10[equiv-lr]* by *simp*
 next
 fix $x v$
 assume $[\langle x^P, H \rangle \text{ in } v]$
 hence 1 : $[\mathcal{A}(\langle x^P, H \rangle) \text{ in } v]$
 using *en-eq-10[equiv-rl]* by *blast*
 have $[x = x \text{ in } v]$
 using *id-eq-1[where 'a= ν]* by *simp*
 hence $[\mathcal{A}(x = x) \text{ in } v]$
 using *id-act-3[equiv-lr]* by *fast*
 hence $[\mathcal{A}(x^P = x^P \ \& \ \langle x^P, H \rangle) \text{ in } v]$
 unfolding *identity- ν -def* using 1 *Act-Basic-2[equiv-rl]* $\&I$ by *blast*
 hence $[\mathcal{A}_\varphi x (x^P) \text{ in } v]$
 unfolding $\varphi\text{-def}$ by *simp*
 moreover have $[\forall \beta. \mathcal{A}_\varphi \beta (x^P) \rightarrow \beta = x \text{ in } v]$
 proof (rule $\forall I$; rule *CP*)
 fix β
 assume $[\mathcal{A}_\varphi \beta (x^P) \text{ in } v]$
 hence $[\mathcal{A}(\beta = x) \text{ in } v]$
 unfolding $\varphi\text{-def}$ *identity- ν -def*
 using *Act-Basic-2[equiv-lr]* $\&E$ by *fast*
 thus $[\beta = x \text{ in } v]$ using *id-act-3[equiv-rl]* by *fast*
 qed
 ultimately have $[\mathcal{A}_\varphi x (x^P) \ \& \ (\forall \beta. \mathcal{A}_\varphi \beta (x^P) \rightarrow \beta = x) \text{ in } v]$
 using $\&I$ by *fast*
 hence $[\exists \alpha. \mathcal{A}_\varphi \alpha (x^P) \ \& \ (\forall \beta. \mathcal{A}_\varphi \beta (x^P) \rightarrow \beta = \alpha) \text{ in } v]$
 by (rule *existential*)
 thus $[\exists !y. \mathcal{A}_\varphi y (x^P) \text{ in } v]$
 unfolding *exists-unique-def* by *simp*
 qed
 have $[\Box(\forall x. \langle F, x^P \rangle \equiv \langle x^P, H \rangle) \text{ in } v]$
 apply (PLM-subst-goal-method
 $\lambda \varphi. \Box(\forall x. \langle F, x^P \rangle \equiv \varphi x)$
 $\lambda x. (\exists !y. \mathcal{A}_\varphi y (x^P))$)
 using 2 $F\text{-def}$ by *auto*
 thus $[\exists F. \Box(\forall x. \langle F, x^P \rangle \equiv \langle x^P, H \rangle) \text{ in } v]$
 by (rule *existential*)
 qed

lemma

assumes *is-propositional*: $(\bigwedge G \varphi. \text{IsProperInX } (\lambda x. \langle G, \iota y. \varphi y x \rangle))$
 and *Abs-x*: $[\langle A!, x^P \rangle \text{ in } v]$
 and *Abs-y*: $[\langle A!, y^P \rangle \text{ in } v]$
 and *noteq*: $[x \neq y \text{ in } v]$
 shows *diffprop*: $[\exists F. \neg(\langle F, x^P \rangle \equiv \langle F, y^P \rangle) \text{ in } v]$
 proof –
 have $[\exists F. \neg(\langle x^P, F \rangle \equiv \langle y^P, F \rangle) \text{ in } v]$

```

    using noteq unfolding exists-def
proof (rule reductio-aa-2)
  assume 1:  $[\forall F. \neg(\llbracket x^P, F \rrbracket \equiv \llbracket y^P, F \rrbracket) \text{ in } v]$ 
  {
    fix F
    have  $[(\llbracket x^P, F \rrbracket \equiv \llbracket y^P, F \rrbracket) \text{ in } v]$ 
      using 1[THEN  $\forall E$ ] useful-tautologies-1[deduction] by blast
  }
  hence  $[\forall F. \llbracket x^P, F \rrbracket \equiv \llbracket y^P, F \rrbracket \text{ in } v]$  by (rule  $\forall I$ )
  thus  $[x = y \text{ in } v]$ 
    unfolding identity- $\nu$ -def
    using ab-obey-1[deduction, deduction]
    Abs-x Abs-y &I by blast
qed
then obtain H where H-def:  $[\neg(\llbracket x^P, H \rrbracket \equiv \llbracket y^P, H \rrbracket) \text{ in } v]$ 
  by (rule Instantiate)
hence 2:  $[(\llbracket x^P, H \rrbracket \ \& \ \neg\llbracket y^P, H \rrbracket) \vee (\neg\llbracket x^P, H \rrbracket \ \& \ \llbracket y^P, H \rrbracket) \text{ in } v]$ 
  apply - by PLM-solver
have  $[\exists F. \Box(\forall x. \llbracket F, x^P \rrbracket \equiv \llbracket x^P, H \rrbracket) \text{ in } v]$ 
  using Fx-equiv-xH[OF is-propositional, THEN  $\forall E$ ] by simp
then obtain F where  $[\Box(\forall x. \llbracket F, x^P \rrbracket \equiv \llbracket x^P, H \rrbracket) \text{ in } v]$ 
  by (rule Instantiate)
hence F-prop:  $[\forall x. \llbracket F, x^P \rrbracket \equiv \llbracket x^P, H \rrbracket \text{ in } v]$ 
  using qml-2[axiom-instance, deduction] by blast
hence a:  $[\llbracket F, x^P \rrbracket \equiv \llbracket x^P, H \rrbracket \text{ in } v]$ 
  using  $\forall E$  by blast
have b:  $[\llbracket F, y^P \rrbracket \equiv \llbracket y^P, H \rrbracket \text{ in } v]$ 
  using F-prop  $\forall E$  by blast
{
  assume 1:  $[\llbracket x^P, H \rrbracket \ \& \ \neg\llbracket y^P, H \rrbracket \text{ in } v]$ 
  hence  $[\llbracket F, x^P \rrbracket \text{ in } v]$ 
    using a[equiv-rl] &E by blast
  moreover have  $[\neg\llbracket F, y^P \rrbracket \text{ in } v]$ 
    using b[THEN oth-class-taut-5-d[equiv-lr], equiv-rl] 1[conj2] by auto
  ultimately have  $[\llbracket F, x^P \rrbracket \ \& \ (\neg\llbracket F, y^P \rrbracket) \text{ in } v]$ 
    by (rule &I)
  hence  $[(\llbracket F, x^P \rrbracket \ \& \ \neg\llbracket F, y^P \rrbracket) \vee (\neg\llbracket F, x^P \rrbracket \ \& \ \llbracket F, y^P \rrbracket) \text{ in } v]$ 
    using  $\vee I$  by blast
  hence  $[\neg(\llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \text{ in } v]$ 
    using oth-class-taut-5-j[equiv-rl] by blast
}
moreover {
  assume 1:  $[\neg\llbracket x^P, H \rrbracket \ \& \ \llbracket y^P, H \rrbracket \text{ in } v]$ 
  hence  $[\llbracket F, y^P \rrbracket \text{ in } v]$ 
    using b[equiv-rl] &E by blast
  moreover have  $[\neg\llbracket F, x^P \rrbracket \text{ in } v]$ 
    using a[THEN oth-class-taut-5-d[equiv-lr], equiv-rl] 1[conj1] by auto
  ultimately have  $[\neg\llbracket F, x^P \rrbracket \ \& \ \llbracket F, y^P \rrbracket \text{ in } v]$ 
    using &I by blast
  hence  $[(\llbracket F, x^P \rrbracket \ \& \ \neg\llbracket F, y^P \rrbracket) \vee (\neg\llbracket F, x^P \rrbracket \ \& \ \llbracket F, y^P \rrbracket) \text{ in } v]$ 
    using  $\vee I$  by blast
  hence  $[\neg(\llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \text{ in } v]$ 
    using oth-class-taut-5-j[equiv-rl] by blast
}
ultimately have  $[\neg(\llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \text{ in } v]$ 
  using 2 intro-elim-4-b reductio-aa-1 by blast
thus  $[\exists F. \neg(\llbracket F, x^P \rrbracket \equiv \llbracket F, y^P \rrbracket) \text{ in } v]$ 
  by (rule existential)
qed

```

lemma original-paradox:

```

assumes is-propositional:  $(\bigwedge G \ \varphi. \text{IsProperInX } (\lambda x. \llbracket G, \iota y. \varphi \ y \ x \rrbracket))$ 
shows False

```

```

proof –
  fix  $v$ 
  have  $[\exists x y. \langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ x \neq y \ \& \ (\forall F. \langle F, x^P \rangle \equiv \langle F, y^P \rangle) \text{ in } v]$ 
    using aclassical2 by auto
  then obtain  $x$  where
     $[\exists y. \langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ x \neq y \ \& \ (\forall F. \langle F, x^P \rangle \equiv \langle F, y^P \rangle) \text{ in } v]$ 
    by (rule Instantiate)
  then obtain  $y$  where xy-def:
     $[\langle A!, x^P \rangle \ \& \ \langle A!, y^P \rangle \ \& \ x \neq y \ \& \ (\forall F. \langle F, x^P \rangle \equiv \langle F, y^P \rangle) \text{ in } v]$ 
    by (rule Instantiate)
  have  $[\exists F. \neg(\langle F, x^P \rangle \equiv \langle F, y^P \rangle) \text{ in } v]$ 
    using diffprop[OF assms, OF xy-def[conj1, conj1, conj1],
      OF xy-def[conj1, conj1, conj2],
      OF xy-def[conj1, conj2]]
    by auto
  then obtain  $F$  where  $[\neg(\langle F, x^P \rangle \equiv \langle F, y^P \rangle) \text{ in } v]$ 
    by (rule Instantiate)
  moreover have  $[\langle F, x^P \rangle \equiv \langle F, y^P \rangle \text{ in } v]$ 
    using xy-def[conj2] by (rule  $\forall E$ )
  ultimately have  $\bigwedge \varphi. [\varphi \text{ in } v]$ 
    using PLM.raa-cor-2 by blast
  thus False
    using Semantics.T4 by auto
qed
end

```