Automation of the Principia Metaphysica in HOL: Part I

Christoph E. Benzmüller and Paul E. Oppenheimer and Edward N. Zalta

December 11, 2015

1 Introduction

We present a formalisation and partial automation of an initial part of the (third authors) Principia Metaphysica [6] in Isabelle/HOL [5].

The Principia Metaphysica, which is based on and extends the Theory of Abstract Objects [**?**], employs a modal relational type theory as logical foundation. Arguments defending this choice against a modal functional type theory have been presented before [7]. In a nutshell, the situation is this: functional type theory comes with strong comprehension principles, which, in the context of the Theory of Abstract Objects, have paradoxical implications. When starting off with a relational foundation, however, weaker comprehension principles are provided, and these obstacles can be avoided.

Isabelle/HOL is a proof assistant based on a functional type theory, more precisely, Church's theory of types [4]. Recently, it has been shown that Church's type theory can be elegantly utilized as a meta-logic to encode and automate various quantified non-classical logics, including modal functional type theory [2, 3]. This work has subsequently been employed in a case study in computational metaphysics, in which different variants of Kurt Gdel's ontological argument [1] were verified (respectively, falsified).

The motivating research questions for the work presented below include:

- Can functional type theory, despite the problems as pointed by Zalta and Oppenheimer, be utilized to encode the Theory of Abstract Objects when following the embeddings approach?
- How elegant and user-friendly is the resulting formalization? In other words, to what extend can Isabelle's user interface be facilitated to hide unpleasant technicalities of the (extended) embedding from the user?
- How far can automation be pushed in the approach? How much user interaction can be avoided in the formalization of the (first part) of the Principia Metaphysica?
- Can the consistency of the theory be validated with the available automated reasoning tools?
- Can the reasoners eventually even contribute some new knowledge? ...

The encoding of modal functional type theory in functional type theory as explored in previous work [2, 3] is simple: modal logic formulas are identified with certain functional type theory

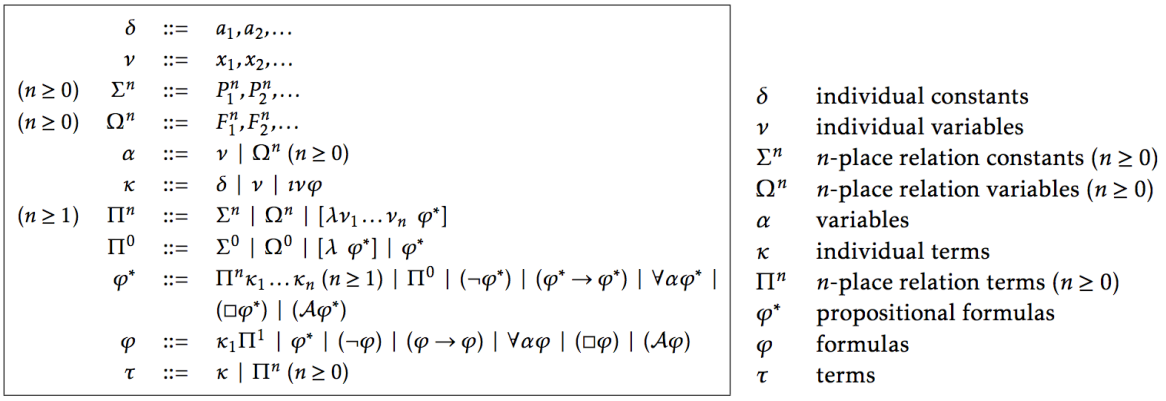

Figure 1: Grammar of Modal Relational Type Theory. Note that two kinds of (complex) formulas are introduced: ones that may have encoding subformulas and ones that dont. The latter are designated as propositional formulas, the former ones simply as formulas.

formulas of predicate type $i \Rightarrow bool$ (abbreviated as io below). Possible worlds are explicitly represented by terms of type i. A modal logic φ formula holds for a world w if and only if the application $\varphi \; w$ evaluates to true. The definition of the propositional modal logic connectives is then straightforward and it simply realizes the standard translation as a set of equations in functional type theory. The approach has been successfully extended for quantifiers. A crucial aspect thereby is that in simple type theory quantifiers can be treated as ordinary logical connectives. No extra binding mechanism is needed since the already existing lambda binding mechanism can be elegantly utilized.

The challenge here is to appropriately 'restrict' this embedding for modal relational type theory.

To achieve this we provide means to explicitly represents and maintain information and constraints on the syntactical structure of modal relational type theory, in particular, we provide means to distinguish between propositional formulas, formulas, terms and erreneous (disallowed) formations. This clearly creates some technical overhead. However, we exploit facilities in Isabelle/HOL's user interface, and other means, to hide most of these technicalities from the user in applications.

2 Preliminaries

We start out with some type declarations and type abbreviations. Our formalism explicitly encodes possible world semantics. Hence, we introduce a distinguished type i to represent the set of possible worlds. Consequently, terms of this type denote possible worlds. Moreover, modal logic formulas are associated in our approach with predicates (resp. sets) on possible worlds. Hence, modal logic formulas have type $(i \Rightarrow bool)$. To make our representation in the remainder more concise we abbreviate this type as io.

typedecl i
type-synonym $io = (i \Rightarrow bool)$

Entities in the abstract theory of types are represented in our formalism by the type e. We

call this the raw type of entities resp. objects. Later on we will introduce means to distinguish between abstract and ordinary entities.

typedecl e

To explicitly model the syntactical restrictions of modal relational type theory we introduce a (polymorphic) datatype $'a\ opt$ (where $'a$ is a polymorphic variable in Isabelle) based on four constructors: $ERR\ 'a$ (identifies erroneous term constructions), $P\ 'a$ (identifies propositional formulas), $F\ 'a$ (identifies formulas), and $T\ 'a$ (identifies terms, such as lambda abstractions). The embeddings approach will be suitably adapted below so that for each language expression (in the embedded modal relational type theory) the respective datatype is identified and appropriately propagated. The encapsulated expressions (the polymorphic type $'a$ will be instantiated below) realize the actual modeling of the logic embedding analogous to previous work for modal functional type theory.

datatype $'a\ opt = ERR\ 'a \mid P\ 'a \mid F\ 'a \mid T\ 'a$

The following operators support a concise and elegant superscript annotation with these four syntactical categories for our language constructs.

abbreviation $mkP{::}io{\Rightarrow}io\ opt\ (\text{-}^{P}\ [109]\ 110)$ **where** $\varphi^{P} \equiv P\ \varphi$
abbreviation $mkF{::}io{\Rightarrow}io\ opt\ (\text{-}^{F}\ [109]\ 110)$ **where** $\varphi^{F} \equiv F\ \varphi$
abbreviation $mkT{::}'a{\Rightarrow}'a\ opt\ (\text{-}^{T}\ [109]\ 110)$ **where** $\varphi^{T} \equiv T\ \varphi$
abbreviation $mkE{::}'a{\Rightarrow}'a\ opt\ (\text{-}^{E}\ [109]\ 110)$ **where** $\varphi^{E} \equiv ERR\ \varphi$

Some language constructs in the theory of abstract types, e.g. the actuality operator $\mathcal{A}$ ("it is actually the case that"), refer to a (fixed) given world. To model such a global world reference we introduce a constant symbol (name) cw of world type i. Moreover, for technical reasons, which will be clarified below, we introduce further (dummy) constant symbols for various domains. Since we assume that all domains are non-empty, introducing these constant symbols is obviously not harmful.

consts $cw :: i$
consts $de{::}e\ dio{::}io\ da{::}'a$

3 Embedding of Modal Relational Type Theory

The language constructs of modal relational type theory are introduced step by step.

The actuality operator $\mathcal{A}$ when applied to a formula or propositional formula φ evaluates φ wrt the fixed given world cw. The compound expression $\mathcal{A}\ \varphi$ inherits its syntactical category F (formula) or P (propositional formula from φ. If the syntactical catagory of φ is ERR (error) or T (term), then the syntactical catagory of $\mathcal{A}\ \varphi$ is ERR and a dummy entity of appropriate type is returned. This illustrates the very idea of our explicit structure and constraints and this scheme will repeated below for all the other language constructs of modal relational type theory.

abbreviation $\mathcal{A}{::}io\ opt \Rightarrow io\ opt$ **where** $\mathcal{A}\ \varphi \equiv case\ \varphi\ of$
$F(\psi) \Rightarrow F(\lambda w.\ \psi\ cw) \mid P(\psi) \Rightarrow P(\lambda w.\ \psi\ cw) \mid \text{-} \Rightarrow ERR(dio)$

The Principia Metaphysica distinguishes between encoding and exemplifying, ... say more ... Encoding $\kappa_1\Pi^1$ is noted here as $\{\!|\kappa_1,\Pi^1|\!\}$. Encoding yields formulas and never propositional formulas. In the embedding encoding is identified with predicate application.

abbreviation *Enc*::*e opt*⇒(*e*⇒*io*) *opt*⇒*io opt* (⦃-,-⦄) **where** ⦃*x*,Φ⦄ ≡ *case* (*x*,Φ) *of*
(*T*(*y*),*T*(*Q*)) ⇒ *F*(λ*w*.(*Q y*) *w*) | - ⇒ *ERR*(*dio*)

Exemplifying formulas $\Pi^1\kappa_1$ are noted here as (⌊Π^1,κ_1⌋). Exemplification yields propositional formulas and never formulas. In the embedding exemplification is identified with predicate application, just as encoding.

abbreviation *Exe1*::(*e*⇒*io*) *opt*⇒*e opt*⇒*io opt* ((⌊-,-⌋)) **where** (⌊Φ,*x*⌋) ≡ *case* (Φ,*x*) *of*
(*T*(*Q*),*T*(*y*)) ⇒ *P*(λ*w*.(*Q y*) *w*) | - ⇒ *ERR*(*dio*)

The Principia Metaphysica supports n-ary exemplification constructions. For pragmatical reasons we consider here the cases only for $n \leq 3$.

abbreviation *Exe2*::(*e*⇒*e*⇒*io*) *opt*⇒*e opt*⇒*e opt*⇒*io opt* ((⌊-,-,-⌋)) **where** (⌊Φ,*x1*,*x2*⌋) ≡ *case* (Φ,*x1*,*x2*) *of*
(*T*(*Q*),*T*(*y1*),*T*(*y2*)) ⇒ *P*(λ*w*.(*Q y1 y2*) *w*) | - ⇒ *ERR*(*dio*)
abbreviation *Exe3*::(*e*⇒*e*⇒*e*⇒*io*) *opt*⇒*e opt*⇒*e opt*⇒*e opt*⇒*io opt* ((⌊-,-,-,-⌋)) **where** (⌊Φ,*x1*,*x2*,*x3*⌋) ≡ *case* (Φ,*x1*,*x2*,*x3*) *of*
(*T*(*Q*),*T*(*y1*),*T*(*y2*),*T*(*y3*)) ⇒ *P*(λ*w*.(*Q y1 y2 y3*) *w*) | - ⇒ *ERR*(*dio*)

Formations with negation and implication are supported for both, formulas and propositional formulas, and their embeddings are straightforward.

abbreviation *not*::*io opt*⇒*io opt* (¬ - [*58*] *59*) **where** ¬ φ ≡ *case* φ *of*
F(ψ) ⇒ *F*(λ*w*.¬(ψ *w*)) | *P*(ψ) ⇒ *P*(λ*w*.¬(ψ *w*)) | - ⇒ *ERR*(*dio*)
abbreviation *implies*::*io opt*⇒*io opt*⇒*io opt* (**infixl** → *51*) **where** φ → ψ ≡ *case* (φ,ψ) *of*
(*F*(α),*F*(β)) ⇒ *F*(λ*w*. α *w* ⟶ β *w*) | (*P*(α),*P*(β)) ⇒ *P*(λ*w*. α *w* ⟶ β *w*) | - ⇒ *ERR*(*dio*)

Also universal quantification ∀ (λ*x*.φ) (first-order and higher-order) is supported for formulas and propositional formulas. Following previous work the embedding maps ∀ (λ*x*.φ) to (λ*w*.∀ *x*.φ*w*). Note that ∀ is introduced as logical connective based on the existing λ-binder. To improve presentation in the remainder we additional introduce binder notation ∀ *x*.φ as syntactic sugar for ∀ (λ*x*.φ).

abbreviation *forall*::('*a*⇒*io opt*)⇒*io opt* (∀) **where** ∀ Φ ≡ *case* (Φ *da*) *of*
F(φ) ⇒ *F*(λ*w*.∀ *x*. *case* (Φ *x*) *of F*(ψ) ⇒ ψ *w*) | *P*(φ) ⇒ *P*(λ*w*.∀ *x*. *case* (Φ *x*) *of P*(ψ) ⇒ ψ *w*) | - ⇒ *ERR*(*dio*)
abbreviation *forallBinder*::('*a*⇒*io opt*)⇒*io opt* (**binder** ∀ [*8*] *9*) **where** ∀ *x*. φ *x* ≡ ∀ φ

The modal □ operator is introduced here for logic S5. Since in an equivalence class of possible worlds each world is reachable from any other world, the guarding accessibility clause in the usual definition of the □ operator can be omitted. This is convenient and should also ease theorem proving. In Section 6.3 we will actually demonstrate that the expected S5 properties are validated by our modeling of □.

abbreviation *box*::*io opt*⇒*io opt* (□ - [*62*] *63*) **where** □φ ≡ *case* φ *of*
F(ψ) ⇒ *F*(λ*w*.∀ *v*. ψ *v*) | *P*(ψ) ⇒ *P*(λ*w*.∀ *v*. ψ *v*) | - ⇒ *ERR*(*dio*)

n-ary lambda abstraction $\boldsymbol{\lambda}^0$,$\boldsymbol{\lambda}$,$\boldsymbol{\lambda}^2$,$\boldsymbol{\lambda}^3$,..., for $n \geq 0$, is supported in the Principia Metaphysica only for propositional formulas. ... say more about $\boldsymbol{\lambda}^0$ ... Their embedding is straightforward: $\boldsymbol{\lambda}^0$ is mapped to identity and $\boldsymbol{\lambda}$,$\boldsymbol{\lambda}^2$,$\boldsymbol{\lambda}^3$,... are mapped to n-ary lambda abstractions, that is, $\boldsymbol{\lambda}(\lambda x.\varphi)$ is mapped to $(\lambda x.\varphi)$ and $\boldsymbol{\lambda}^2(\lambda xy.\varphi)$ to $(\lambda xy.\varphi)$. For pragmatical reasons we restrict ourselves here to the cases where $n \leq 0$. Binder notation is introduced for $\boldsymbol{\lambda}$ (... unfortuntaley, I don't know yet how this can be achieved as well for $\boldsymbol{\lambda}^2$,$\boldsymbol{\lambda}^3$,... ...)

abbreviation *lam0*::*io opt*⇒*io opt* (**λ**^{0}) **where** **λ**$^0\varphi \equiv$ *case* φ *of*
$P(\psi) \Rightarrow P(\psi)$ | - ⇒ *ERR dio*

abbreviation *lam*::(*e*⇒*io opt*)⇒(*e*⇒*io*) *opt* (**λ**) **where** **λ**Φ ≡ *case* (Φ *de*) *of*
$P(\varphi) \Rightarrow T(\lambda x.$ *case* (Φ *x*) *of* $P(\varphi) \Rightarrow \varphi)$ | - ⇒ *ERR*($\lambda x.$ *dio*)
abbreviation *lamBinder*::(*e*⇒*io opt*)⇒(*e*⇒*io*) *opt* (**binder** **λ** [*8*] *9*) **where** **λ** $x.\ \varphi\ x \equiv$ **λ** φ

abbreviation *lam2*::(*e*⇒*e*⇒*io opt*)⇒(*e*⇒*e*⇒*io*) *opt* (**λ**^{2}) **where** **λ**^{2}Φ ≡ *case* (Φ *de de*) *of*
$P(\varphi) \Rightarrow T(\lambda x\ y.$ *case* (Φ *x y*) *of* $P(\varphi) \Rightarrow \varphi)$ | - ⇒ *ERR*($\lambda x\ y.$ *dio*)

abbreviation *lam3*::(*e*⇒*e*⇒*e*⇒*io opt*)⇒(*e*⇒*e*⇒*e*⇒*io*) *opt* (**λ**^{3}) **where** **λ**^{3}Φ ≡ *case* (Φ *de de de*) *of*
$P(\varphi) \Rightarrow T(\lambda x\ y\ z.$ *case* (Φ *x y z*) *of* $P(\varphi) \Rightarrow \varphi)$ | - ⇒ *ERR*($\lambda x\ y\ z.$ *dio*)

The Principia Metaphysica supports rigid definite descriptions. Our definition maps $\iota(\lambda x.\varphi)$ to (*THE* $x.\ \varphi$ *cw*), that is Isabelle's inbuilt description operator THE is utilized and evaluation is rigidly carried out with respect to the current world *cw*. Moreover, application of that is allowed only if the body of Φ, computed by clause (Φ *de*), is a propositional formula. In this case a term is returned and otherwise error reported (for dummy entity *de*). We again introduce binder notation.

abbreviation *that*::(*e*⇒*io opt*)⇒*e opt* (*ι*) **where** *ι* Φ ≡ *case* (Φ *de*) *of*
$P(\varphi) \Rightarrow T($*THE* $x.$ *case* (Φ *x*) *of* $P\ \psi \Rightarrow \psi$ *cw*) | - ⇒ *ERR*(*de*)
abbreviation *thatBinder*::(*e*⇒*io opt*)⇒*e opt* (**binder** *ι* [*8*] *9*) **where** *ι* $x.\ \varphi\ x \equiv \iota\ \varphi$

4 Further Logical Connectives

Further logical connectives can be defined as usual. For existential quantification we here prefer a native introduction, even though a definition based on ¬ and ∀ is also possible (but syntactically not more elegant). For pragmatic reasons it eventually makes sense to to prefer native introductions for all connectives.

abbreviation *conj*::*io opt*⇒*io opt*⇒*io opt* (**infixl** ∧ *53*) **where** $\varphi\wedge\psi \equiv \neg(\varphi\rightarrow\neg\psi)$
abbreviation *or*::*io opt*⇒*io opt*⇒*io opt* (**infixl** ∨ *52*) **where** $\varphi\vee\psi \equiv \neg\varphi\rightarrow\psi$
abbreviation *equivalent*::*io opt*⇒*io opt*⇒*io opt* (**infixl** ≡ *51*) **where** $\varphi\equiv\psi \equiv (\varphi\rightarrow\psi)\wedge(\psi\rightarrow\varphi)$
abbreviation *diamond*::*io opt*⇒*io opt* (◇ - [*62*] *63*) **where** $\Diamond\varphi \equiv \neg\Box(\neg\varphi)$
abbreviation *exists*::('*a*⇒*io opt*)⇒*io opt* (∃) **where** ∃ Φ ≡ *case* (Φ *da*) *of*
$P\ \varphi \Rightarrow P(\lambda w.\exists x.$ *case* (Φ *x*) *of* $P\ \psi \Rightarrow \psi\ w)$ | $F\ \varphi \Rightarrow F(\lambda w.\ \exists x.$ *case* (Φ *x*) *of* $F\ \psi \Rightarrow \psi\ w)$ | - ⇒ *ERR dio*
abbreviation *existsBinder*::('*a*⇒*io opt*)⇒*io opt* (**binder** ∃ [*8*] *9*) **where** ∃ $x.\ \varphi\ x \equiv$ ∃ φ

5 Meta-Logic

Our approach to rigorously distinguish between proper and improper language constructions and to explicitly maintain respective information is continued also at meta-level. For this we introduce three truth values *tt*, *ff* and *err*, representing truth, falsity and error. These values can also be noted as ⊤, ⊥ and ∗. We could, of course, also introduce respective logical connectives for the meta-level, but in our applications so far (see below) this was not yet relevant.

datatype *mf* = *tt* ($\top$) | *ff* ($\bot$) | *err* ($*$)

Next we define the meta-logical notions of validity, satisfiability, countersatisfiability and invalidity for our embedded modal relational type theory. To support concise formula representations in the remainder we introduce the following notations: $[\varphi]$ (for φ is valid), $[\varphi]^{sat}$ (φ is satisfiability), $[\varphi]^{csat}$ (φ is countersatisfiability) and $[\varphi]^{inv}$ (φ is invalid).

abbreviation *valid* :: *io opt⇒mf* ([-] [*1*]) **where** $[\varphi] \equiv$ *case* φ *of*
$P(\psi) \Rightarrow$ *if* $\forall w.(\psi\ w) \longleftrightarrow$ *True then* $\top$ *else* $\bot$ | $F(\psi) \Rightarrow$ *if* $\forall w.(\psi\ w) \longleftrightarrow$ *True then* $\top$ *else* $\bot$ | - $\Rightarrow *$
abbreviation *satisfiable* :: *io opt⇒mf* ([-]^{sat} [*1*]) **where** $[\varphi]^{sat} \equiv$ *case* φ *of*
$P(\psi) \Rightarrow$ *if* $\exists w.(\psi\ w) \longleftrightarrow$ *True then* $\top$ *else* $\bot$ | $F(\psi) \Rightarrow$ *if* $\exists w.(\psi\ w) \longleftrightarrow$ *True then* $\top$ *else* $\bot$ | - $\Rightarrow *$
abbreviation *countersatisfiable* :: *io opt⇒mf* ([-]^{csat} [*1*]) **where** $[\varphi]^{csat} \equiv$ *case* φ *of*
$P(\psi) \Rightarrow$ *if* $\exists w.\neg(\psi\ w) \longleftrightarrow$ *True then* $\top$ *else* $\bot$ | $F(\psi) \Rightarrow$ *if* $\exists w.\neg(\psi\ w) \longleftrightarrow$ *True then* $\top$ *else* $\bot$ | - $\Rightarrow *$
abbreviation *invalid* :: *io opt⇒mf* ([-]^{inv} [*1*]) **where** $[\varphi]^{inv} \equiv$ *case* φ *of*
$P(\psi) \Rightarrow$ *if* $\forall w.\neg(\psi\ w) \longleftrightarrow$ *True then* $\top$ *else* $\bot$ | $F(\psi) \Rightarrow$ *if* $\forall w.\neg(\psi\ w) \longleftrightarrow$ *True then* $\top$ *else* $\bot$ | - $\Rightarrow *$

6 Some Basic Tests

6.1 Verifying Necessitation

The next two lemmata show that necessitation holds for arbitrary formulas and arbitrary propositional formulas. We present the lemma in both variants.

lemma *necessitationF*: $[\varphi^F] = \top \longrightarrow [\Box\varphi^F] = \top$ **apply** *simp* **done**
lemma *necessitationP*: $[\varphi^P] = \top \longrightarrow [\Box\varphi^P] = \top$ **apply** *simp* **done**

6.2 Modal Collapse is Countersatisfiable

The modelfinder Nitpick constructs a finite countermodel to the assertion that modal collaps is valid. Nitpicks countermodel consists of four worlds i1, i2, i3 and i4. Moreover, it defines φ to hold only in world i3 and it suggests i3 as the actual world in which to evaluate the conjecture formula. This countermodel is not minimal.

lemma *modalCollapseF*: $[\varphi^F \rightarrow \Box\varphi^F] = \top$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick
lemma *modalCollapseP*: $[\varphi^P \rightarrow \Box\varphi^P] = \top$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick

6.3 Verifying S5 Principles

$\Box$ could have been modeled by employing an equivalence relation r in a guarding clause. This has been done in previous work. In a simpler, alternative approach we omit this clause here (since all worlds are reachable from any world in an equivalence relation). The following lemmata, which check various conditions for S5, ensure that we have indeed obtain a correct modeling of S5.

lemma *axiom-T-P*: $[\Box\varphi^P \rightarrow \varphi^P] = \top$ **apply** *simp* **done**
lemma *axiom-T-F*: $[\Box\varphi^F \rightarrow \varphi^F] = \top$ **apply** *simp* **done**

lemma *axiom-B-P*: $[\varphi^P \rightarrow \Box\Diamond\varphi^P] = \top$ **apply** *simp* **done**
lemma *axiom-B-F*: $[\varphi^F \rightarrow \Box\Diamond\varphi^F] = \top$ **apply** *simp* **done**

lemma *axiom-4-P*: $[\Box\varphi^P \rightarrow \Diamond\varphi^P] = \top$ **apply** *simp* **by** *auto*
lemma *axiom-4-F*: $[\Box\varphi^F \rightarrow \Diamond\varphi^F] = \top$ **apply** *simp* **by** *auto*

lemma *axiom-D-P*: $[\Box\varphi^P \rightarrow \Box\Box\varphi^P] = \top$ **apply** *simp* **done**
lemma *axiom-D-F*: $[\Box\varphi^F \rightarrow \Box\Box\varphi^F] = \top$ **apply** *simp* **done**

lemma *axiom-5-P*: $[\Diamond\varphi^P \rightarrow \Box\Diamond\varphi^P] = \top$ **apply** *simp* **done**
lemma *axiom-5-F*: $[\Diamond\varphi^F \rightarrow \Box\Diamond\varphi^F] = \top$ **apply** *simp* **done**

lemma *test-A-P*: $[\Box\Diamond\varphi^P \rightarrow \Diamond\varphi^P] = \top$ **apply** *simp* **done**
lemma *test-A-F*: $[\Box\Diamond\varphi^F \rightarrow \Diamond\varphi^F] = \top$ **apply** *simp* **done**

lemma *test-B-P*: $[\Diamond\Box\varphi^P \rightarrow \Diamond\varphi^P] = \top$ **apply** *simp* **by** *auto*
lemma *test-B-F*: $[\Diamond\Box\varphi^F \rightarrow \Diamond\varphi^F] = \top$ **apply** *simp* **by** *auto*

lemma *test-C-P*: $[\Box\Diamond\varphi^P \rightarrow \Box\varphi^P] = \top$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick
lemma *test-C-F*: $[\Box\Diamond\varphi^F \rightarrow \Box\varphi^F] = \top$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick

lemma *test-D-P*: $[\Diamond\Box\varphi^P \rightarrow \Box\varphi^P] = \top$ **apply** *simp* **done**
lemma *test-D-F*: $[\Diamond\Box\varphi^F \rightarrow \Box\varphi^F] = \top$ **apply** *simp* **done**

6.4 Relations between Meta-Logical Notions

lemma $[\varphi^P] = \top \longleftrightarrow [\varphi^P]^{csat} = \bot$ **apply** *simp* **done**
lemma $[\varphi^P]^{sat} = \top \longleftrightarrow [\varphi^P]^{inv} = \bot$ **apply** *simp* **done**
lemma $[\varphi^F] = \top \longleftrightarrow [\varphi^F]^{csat} = \bot$ **apply** *simp* **done**
lemma $[\varphi^F]^{sat} = \top \longleftrightarrow [\varphi^F]^{inv} = \bot$ **apply** *simp* **done**

However, for terms we have

lemma $[\varphi^T] = *$ **apply** *simp* **done**
lemma $[\varphi^T]^{sat} = *$ **apply** *simp* **done**
lemma $[\varphi^T]^{csat} = *$ **apply** *simp* **done**
lemma $[\varphi^T]^{inv} = *$ **apply** *simp* **done**

6.5 Testing for the Correct Propagation of Syntactical Category Information

lemma $\exists\, X.\ (\!|R^T,a^T|\!) = X^P \wedge \neg(\exists\, X.\ (\!|R^T,a^T|\!) = X^F) \wedge \neg(\exists\, X.\ (\!|R^T,a^T|\!) = X^T) \wedge \neg(\exists\, X.\ (\!|R^T,a^T|\!) = X^E)$ **apply** *simp* **done**
lemma $\exists\, X.\ \{\!|x^T,R^T|\!\} = X^F \wedge \neg(\exists\, X.\ \{\!|x^T,R^T|\!\} = X^P) \wedge \neg(\exists\, X.\ \{\!|x^T,R^T|\!\} = X^T) \wedge \neg(\exists\, X.\ \{\!|x^T,R^T|\!\} = X^E)$ **apply** *simp* **done**

Most importantly, we have that the following language constructing is evaluated as erroneous at validity level.

lemma $[(\!|\boldsymbol{\lambda}x.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\},a^T|\!)] = *$ **apply** *simp* **done**

This is also confirmed as follows in Isabelle: Isabelle simplifies the following expression to $dio^E = X$ (simply move the curse on *simp* to see this).

Todo: ... select, adapt, and explain some of examples below ...

lemma $(\!|\boldsymbol{\lambda}x.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\},a^T|\!) = X$ **apply** *simp* **oops**

lemma $\exists\, X.\ (\!|\boldsymbol{\lambda} x.\ (\!|R^T,x^T|\!) \rightarrow (\!|R^T,x^T|\!),a^T|\!) = X^P$ **apply** *simp* **done** — Solution: $(\lambda w.\ \mathit{True}) = X$

lemma $[(\!|\boldsymbol{\lambda} x.\ (\!|R^T,x^T|\!) \rightarrow (\!|R^T,x^T|\!),a^T|\!)] = \top$ **apply** *simp* **done**

lemma $(\!|\boldsymbol{\lambda} x.\ (\!|R^T,x^T|\!) \rightarrow (\!|R^T,x^T|\!),a^T|\!) = X^P$ **apply** *simp* **oops**
lemma $(\boldsymbol{\lambda} x.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\}) = X$ **apply** *simp* **oops**

lemma $[(\forall\, x.\ (\!|R^T,x^T|\!) \rightarrow (\!|R^T,x^T|\!))] = \top$ **apply** *simp* **done**
lemma $[(\forall\, R.\ \forall\, (\lambda x.\ (\!|R^T,x^T|\!) \rightarrow (\!|R^T,x^T|\!)))] = \top$ **apply** *simp* **done**
lemma $(\forall\, x.\ (\!|R^T,x^T|\!) \rightarrow (\!|R^T,x^T|\!)) = X$ **apply** *simp* **oops**

lemma $[(\forall\, x.\ \{\!|x^T,R^T|\!\} \rightarrow \{\!|x^T,R^T|\!\})] = \top$ **apply** *simp* **done**
lemma $(\forall\, x.\ \{\!|x^T,R^T|\!\} \rightarrow \{\!|x^T,R^T|\!\}) = X$ **apply** *simp* **oops**

lemma $[(\exists\, x\ y.\ \{\!|x^T,R^T|\!\} \rightarrow \{\!|y^T,R^T|\!\})] = \top$ **apply** *simp* **done**

lemma $[(\forall\, x.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\})] = *$ **apply** *simp* **done**
lemma $[(\forall\, x.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\})] = X$ **apply** *simp* **oops**
lemma $(\forall\, x.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\}) = X$ **apply** *simp* **oops**
lemma $[(\forall\, R.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\})] = *$ **apply** *simp* **done**
lemma $(\forall\, R.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\}) = X$ **apply** *simp* **oops**

lemma $[(\forall\, x.\ \exists\, (\lambda R.\ \{\!|x^T,R^T|\!\} \rightarrow \{\!|x^T,R^T|\!\}))] = \top$ **apply** *simp* **done**
lemma $[(\exists\, x.\ \forall\, (\lambda R.\ \{\!|x^T,R^T|\!\} \rightarrow \{\!|x^T,R^T|\!\}))] = \top$ **apply** *simp* **done**

6.6 Are Priorities Defined Correctly?

lemma $\varphi^P \wedge \psi^P \rightarrow \chi^P \equiv (\varphi^P \wedge \psi^P) \rightarrow \chi^P$ **apply** *simp* **done**
lemma $\varphi^P \wedge \psi^P \rightarrow \chi^P \equiv \varphi^P \wedge (\psi^P \rightarrow \chi^P)$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick

lemma $(\varphi^P \wedge \psi^P \equiv \varphi^P \wedge \psi^P) \equiv ((\varphi^P \wedge \psi^P) \equiv (\varphi^P \wedge \psi^P))$ **apply** *simp* **done**
lemma $(\varphi^P \wedge \psi^P \equiv \varphi^P \wedge \psi^P) \equiv (\varphi^P \wedge (\psi^P \equiv \varphi^P) \wedge \psi^P)$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick

7 E!, O!, A! and =E

We introduce the distinguished 1-place relation constant: E (read: being concrete or concreteness)

consts E::$(e \Rightarrow io)$

Being ordinary is defined as being possibly concrete.

abbreviation *ordinaryObject*::$(e \Rightarrow io)$ *opt* $(O!)$ **where** $O! \equiv \boldsymbol{\lambda} x.\ \Diamond(\!|E^T,x^T|\!)$

Being abstract is is defined as not possibly being concrete.

abbreviation *abstractObject*::$(e \Rightarrow io)$ *opt* $(A!)$ **where** $A! \equiv \boldsymbol{\lambda} x.\ \neg(\Diamond(\!|E^T,x^T|\!))$

Identity relations $=_E$ and $=$ are introduced.

abbreviation *identityE*::*e opt*⇒*e opt*⇒*io opt* (**infixl** $=_E$ *63*)

where $x =_E y \equiv (\!|O!,x|\!) \wedge (\!|O!,y|\!) \wedge \Box(\forall F.\ (\!|F^T,x|\!) \equiv (\!|F^T,y|\!))$

abbreviation *identity*::*e opt*⇒*e opt*⇒*io opt* (**infixl** = *63*)
where $x = y \equiv (x =_E y) \vee ((\!|A!,x|\!) \wedge (\!|A!,y|\!) \wedge \Box(\forall F.\ \{\!|x,F^T|\!\} \equiv \{\!|y,F^T|\!\}))$

7.1 Further Test Examples

Todo: ... select, adapt, and explain some of examples below

lemma $[a^T =_E a^T] = \top$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick

lemma $[(\!|O!,a^T|\!) \rightarrow a^T =_E a^T] = \top$ **apply** *simp* **done**

lemma $[(\forall F.\ (\!|F^T,x^T|\!) \equiv (\!|F^T,x^T|\!))] = \top$ **apply** *simp* **done**
lemma $[(\!|O!,a^T|\!) \rightarrow (\!|\boldsymbol{\lambda}x.\ x^T =_E a^T,a^T|\!)] = \top$ **apply** *simp* **done**

lemma $[(\exists F.\ \{\!|a^T,F^T|\!\})] = \top$ **apply** *simp* **by** *auto*

lemma $[(\exists \varphi.\ \varphi^P)] = \top$ **apply** *simp* **by** *auto*
lemma $[(\exists \varphi.\ \varphi^F)] = \top$ **apply** *simp* **by** *auto*

8 Axioms

8.1 Axioms for Negations and Conditionals

lemma *a21-1-P*: $[\varphi^P \rightarrow (\varphi^P \rightarrow \varphi^P)] = \top$ **apply** *simp* **done**
lemma *a21-1-F*: $[\varphi^F \rightarrow (\varphi^F \rightarrow \varphi^F)] = \top$ **apply** *simp* **done**
lemma *a21-2-P*: $[(\varphi^P \rightarrow (\psi^P \rightarrow \chi^P)) \rightarrow ((\varphi^P \rightarrow \psi^P) \rightarrow (\varphi^P \rightarrow \chi^P))] = \top$ **apply** *simp* **done**
lemma *a21-2-F*: $[(\varphi^F \rightarrow (\psi^F \rightarrow \chi^F)) \rightarrow ((\varphi^F \rightarrow \psi^F) \rightarrow (\varphi^F \rightarrow \chi^F))] = \top$ **apply** *simp* **done**
lemma *a21-3-P*: $[(\neg\varphi^P \rightarrow \neg\psi^P) \rightarrow ((\neg\varphi^P \rightarrow \psi^P) \rightarrow \varphi^P)] = \top$ **apply** *simp* **done**
lemma *a21-3-F*: $[(\neg\varphi^F \rightarrow \neg\psi^F) \rightarrow ((\neg\varphi^F \rightarrow \psi^F) \rightarrow \varphi^F)] = \top$ **apply** *simp* **done**

8.2 Axioms of Identity

todo

8.3 Axioms of Quantification

todo

8.4 Axioms of Actuality

lemma *a31-1-P*: $[\mathcal{A}(\neg\varphi^P) \equiv \neg\mathcal{A}(\varphi^P)] = \top$ **apply** *simp* **done**
lemma *a31-1-F*: $[\mathcal{A}(\neg\varphi^F) \equiv \neg\mathcal{A}(\varphi^F)] = \top$ **apply** *simp* **done**
lemma *a31-2-P*: $[\mathcal{A}(\varphi^P \rightarrow \psi^P) \equiv (\mathcal{A}(\varphi^P) \rightarrow \mathcal{A}(\psi^P))] = \top$ **apply** *simp* **done**
lemma *a31-2-F*: $[\mathcal{A}(\varphi^F \rightarrow \psi^F) \equiv (\mathcal{A}(\varphi^F) \rightarrow \mathcal{A}(\psi^F))] = \top$ **apply** *simp* **done**
lemma *a31-3-P*: $[\mathcal{A}(\forall x.\ \varphi^P) \equiv (\forall x.\ \mathcal{A}(\varphi^P))] = \top$ **apply** *simp* **done**
lemma *a31-3-F*: $[(\mathcal{A}(\forall x.\ \varphi^F) \equiv (\forall x.\ \mathcal{A}(\varphi^F)))] = \top$ **apply** *simp* **done**
lemma *a31-4-P*: $[\mathcal{A}(\varphi^P) \equiv \mathcal{A}(\mathcal{A}(\varphi^P))] = \top$ **apply** *simp* **done**
lemma *a31-4-F*: $[\mathcal{A}(\varphi^F) \equiv \mathcal{A}(\mathcal{A}(\varphi^F))] = \top$ **apply** *simp* **done**

8.5 Axioms of Necessity

lemma *a32-1-P*: $[(\Box(\varphi^P \rightarrow \varphi^P)) \rightarrow (\Box\varphi^P \rightarrow \Box\varphi^P)] = \top$ **apply** *simp* **done**
lemma *a32-1-F*: $[(\Box(\varphi^F \rightarrow \varphi^F)) \rightarrow (\Box\varphi^F \rightarrow \Box\varphi^F)] = \top$ **apply** *simp* **done**
lemma *a32-2-P*: $[\Box\varphi^P \rightarrow \varphi^P] = \top$ **apply** *simp* **done**
lemma *a32-2-F*: $[\Box\varphi^F \rightarrow \varphi^F] = \top$ **apply** *simp* **done**

lemma *a32-3-P*: $[\Box\Diamond\varphi^P \rightarrow \Diamond\varphi^P] = \top$ **apply** *simp* **done**
lemma *a32-3-F*: $[\Box\Diamond\varphi^F \rightarrow \Diamond\varphi^F] = \top$ **apply** *simp* **done**
lemma *a32-4-P*: $[(\forall\, x.\ \Box\varphi^P) \rightarrow \Box((\forall\, x.\ \varphi^P))] = \top$ **apply** *simp* **done**
lemma *a32-4-F*: $[(\forall\, x.\ \Box\varphi^F) \rightarrow \Box((\forall\, x.\ \varphi^F))] = \top$ **apply** *simp* **done**

The following needs to be an axiom; it does not follow for free: it is possible that there are contingently concrete individuals and it is possible that there are not:

axiomatization where
a32-5-P: $[\Diamond(\exists\, x.\ (\!|E^T,x^T|\!) \wedge \Diamond(\neg(\!|E^T,x^T|\!))) \wedge \Diamond(\neg(\exists\, x.\ (\!|E^T,x^T|\!) \wedge \Diamond(\neg(\!|E^T,x^T|\!))))] = \top$

A brief check that this axiom is well-formed, i.e. does not return error

lemma $[\Diamond(\exists\, x.\ (\!|E^T,x^T|\!) \wedge \Diamond(\neg(\!|E^T,x^T|\!))) \wedge \Diamond(\neg(\exists\, x.\ (\!|E^T,x^T|\!) \wedge \Diamond(\neg(\!|E^T,x^T|\!))))] \neq *$ **apply** *simp* **done**

8.6 Axioms of Necessity and Actuality

lemma *a33-1-P*: $[\mathcal{A}(\varphi^P) \rightarrow \Box\mathcal{A}(\varphi^P)] = \top$ **apply** *simp* **done**
lemma *a33-1-F*: $[\mathcal{A}(\varphi^F) \rightarrow \Box\mathcal{A}(\varphi^F)] = \top$ **apply** *simp* **done**
lemma *a33-2-P*: $[\Box\varphi^P \equiv \mathcal{A}(\Box\varphi^P)] = \top$ **apply** *simp* **done**
lemma *a33-2-F*: $[\Box\varphi^F \equiv \mathcal{A}(\Box\varphi^F)] = \top$ **apply** *simp* **done**

8.7 Axioms for Descriptions

... here I still run into some problems, need to analyze why both formulas result in errors, check carefully again definition of equality and description ...

lemma *a34-Inst-1*: $[(x^T = (\iota\ x.\ (\!|R^T,x^T|\!))) \equiv (\forall\ z.\ (\mathcal{A}(\!|R^T,z^T|\!) \equiv (z^T = x^T)))] = *$ **apply** *simp* **done**
lemma *a34-Inst-2*: $[(x^T = (\iota\ x.\ (\varphi(x))^T)) \equiv (\forall\ z.\ (\mathcal{A}((\varphi(z))^T) \equiv (z^T = x^T)))] = *$ **apply** *simp* **done**

References

[1] C. Benzmüller and B. W. Paleo. Automating Gödel's ontological proof of God's existence with higher-order automated theorem provers. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *ECAI 2014*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 93 – 98. IOS Press, 2014.

[2] C. Benzmüller and L. Paulson. Quantified multimodal logics in simple type theory. *Logica Universalis (Special Issue on Multimodal Logics)*, 7(1):7–20, 2013.

[3] C. Benzmüller and B. Woltzenlogel Paleo. Automating Gödel's ontological proof of God's existence with higher-order automated theorem provers. In T. Schaub, G. Friedrich, and B. O'Sullivan, editors, *ECAI 2014*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 93 – 98. IOS Press, 2014.

[4] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[5] T. Nipkow, L. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Number 2283 in LNCS. Springer, 2002.

[6] E. N. Zalta. Principia metaphysica, a compilation of the theorems of the theory of abstract objects. Available at `https://mally.stanford.edu/publications.html`.

[7] E. N. Zalta and P. E. Oppenheimer. Relations versus functions at the foundations of logic: Type-theoretic considerations. *Journal of Logic and Computation*, (21):351374, 2011.