Automation of the Principia Metaphysica in HOL: Part I

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

February 17, 2016

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 pointed at 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? 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?
- Suggestions for improvements in Isabelle? Any particular problems detected in the course of the study? . . .

	δ	::=	$a_1, a_2, \ldots$
	ν	::=	$x_1, x_2, \ldots$
$(n \geq 0)$	Σ^n	::=	$P_1^n, P_2^n, \ldots$
$(n \geq 0)$	Ω^n	::=	$F_1^n, F_2^n, \ldots$
	α	::=	$\nu \mid \Omega^n \; (n \geq 0)$
	κ	::=	$\delta \mid \nu \mid \iota\nu\varphi$
$(n \geq 1)$	Π^n	::=	$\Sigma^n \mid \Omega^n \mid [\lambda\nu_1 \ldots \nu_n \; \varphi^*]$
	Π^0	::=	$\Sigma^0 \mid \Omega^0 \mid [\lambda \; \varphi^*] \mid \varphi^*$
	φ^*	::=	$\Pi^n\kappa_1 \ldots \kappa_n \; (n \geq 1) \mid \Pi^0 \mid (\neg\varphi^*) \mid (\varphi^* \to \varphi^*) \mid \forall\alpha\varphi^* \mid (\Box\varphi^*) \mid (\mathcal{A}\varphi^*)$
	φ	::=	$\kappa_1\Pi^1 \mid \varphi^* \mid (\neg\varphi) \mid (\varphi \to \varphi) \mid \forall\alpha\varphi \mid (\Box\varphi) \mid (\mathcal{A}\varphi)$
	τ	::=	$\kappa \mid \Pi^n \; (n \geq 0)$

δ	individual constants
ν	individual variables
Σ^n	n-place relation constants $(n \geq 0)$
Ω^n	n-place relation variables $(n \geq 0)$
α	variables
κ	individual terms
Π^n	n-place relation terms $(n \geq 0)$
φ^*	propositional formulas
φ	formulas
τ	terms

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 do not. The latter are designated as propositional formulas, the former ones simply as formulas.

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 formulas of predicate type $i \Rightarrow bool$ (abbreviated as io below). Possible worlds are explicitly represented by terms of type i. A modal logic formula φ 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 ineligible/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 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 Principia Metaphysica, e.g. the actuality operator $\boldsymbol{\mathcal{A}}$ ("it is actually the case that"), refer to a (fixed) designated world. To model such a rigid dependence we introduce a constant symbol (name) dw 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 $dw :: i$
consts $de{::}e\ dio{::}io\ deio{::}e{\Rightarrow}io\ da{::}'a$

3 Embedding of Modal Relational Type Theory

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

The actuality operator $\boldsymbol{\mathcal{A}}$ when applied to a formula or propositional formula φ evaluates φ wrt the fixed given world cw. The compound expression $\boldsymbol{\mathcal{A}}\varphi$ 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 $\boldsymbol{\mathcal{A}}\varphi$ 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 $Actual{::}io\ opt \Rightarrow io\ opt\ (\boldsymbol{\mathcal{A}}\ \text{-}\ [64]\ 65)$ **where** $\boldsymbol{\mathcal{A}}\varphi \equiv case\ \varphi\ of$
$F(\psi) \Rightarrow F(\lambda w.\ \psi\ dw) \mid P(\psi) \Rightarrow P(\lambda w.\ \psi\ dw) \mid \text{-} \Rightarrow ERR(dio)$

The Principia Metaphysica distinguishes between encoding $\kappa_1\Pi^1$ and exemplifying Π^n,κ_1,..,κ_n. ... say more ... Exemplification is supported here only for $1{\leq}n{\leq}3$.

First we introduce the primitive constant *enc*. This basic primitives is employed below in the definition of the encoding operation $\{\!|\kappa_1,\Pi^1|\!\}$.

consts $enc{::}e{\Rightarrow}(e{\Rightarrow}io){\Rightarrow}io$

Encoding $\kappa_1\Pi^1$ is noted below as $\{\!|\kappa_1,\Pi^1|\!\}$. Encoding yields formulas and never propositional formulas.

abbreviation $Enc{::}e\ opt{\Rightarrow}(e{\Rightarrow}io)\ opt{\Rightarrow}io\ opt\ (\{\!|\text{-},\text{-}|\!\})$ **where** $\{\!|x,\Phi|\!\} \equiv case\ (x,\Phi)\ of$
$\quad (T(y),T(Q)) \Rightarrow F(enc\ y\ Q) \mid \text{-} \Rightarrow ERR(dio)$

Exemplifying formulas $\Pi^1\kappa_1$ are noted here as $(|\Pi^1,\kappa_1|)$. Exemplification yields propositional formulas. Exemplification is mapped to predicate application.

abbreviation $Exe1{::}(e{\Rightarrow}io)\ opt{\Rightarrow}e\ opt{\Rightarrow}io\ opt\ ((|\text{-},\text{-}|))$ **where** $(|\Phi,x|) \equiv case\ (\Phi,x)\ of$
$\quad (T(Q),T(y)) \Rightarrow P(Q\ y) \mid \text{-} \Rightarrow ERR(dio)$

The Principia Metaphysica supports n-ary exemplification constructions. We support the cases for $1{\leq}n{\leq}3$. Exemplification is mapped to predicate application.

abbreviation $Exe2{::}(e{\Rightarrow}e{\Rightarrow}io)\ opt{\Rightarrow}e\ opt{\Rightarrow}e\ opt{\Rightarrow}io\ opt\ ((|\text{-},\text{-},\text{-}|))$
where $(|\Phi,x1,x2|) \equiv case\ (\Phi,x1,x2)\ of$
$\quad (T(Q),T(y1),T(y2)) \Rightarrow P(Q\ y1\ y2) \mid \text{-} \Rightarrow ERR(dio)$
abbreviation $Exe3{::}(e{\Rightarrow}e{\Rightarrow}e{\Rightarrow}io)\ opt{\Rightarrow}e\ opt{\Rightarrow}e\ opt{\Rightarrow}e\ opt{\Rightarrow}io\ opt\ ((|\text{-},\text{-},\text{-},\text{-}|))$
where $(|\Phi,x1,x2,x3|) \equiv case\ (\Phi,x1,x2,x3)\ of$
$\quad (T(Q),T(y1),T(y2),T(y3)) \Rightarrow P(Q\ y1\ y2\ y3) \mid \text{-} \Rightarrow ERR(dio)$

Formations with negation and implication are supported for both, formulas and propositional formulas, and their embeddings are straightforward. In the case of implication the compound formula is a propositional formula only of both subformulas are propositional formulas. If at one is a formula and the other one eligible, then the compound formula is a formula. In all other cases an ERR-Formula is returned.

abbreviation $not{::}io\ opt{\Rightarrow}io\ opt\ (\neg\ \text{-}\ [58]\ 59)$ **where** $\neg\ \varphi \equiv case\ \varphi\ of$
$\quad F(\psi) \Rightarrow F(\lambda w.\neg(\psi\ w)) \mid P(\psi) \Rightarrow P(\lambda w.\neg(\psi\ w)) \mid \text{-} \Rightarrow ERR(dio)$
abbreviation $implies{::}io\ opt{\Rightarrow}io\ opt{\Rightarrow}io\ opt$ (**infixl** $\rightarrow$ 51) **where** $\varphi \rightarrow \psi \equiv case\ (\varphi,\psi)\ of$
$\quad (P(\alpha),P(\beta)) \Rightarrow P(\lambda w.\ \alpha\ w \longrightarrow \beta\ w) \mid (F(\alpha),F(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \longrightarrow \beta\ w) \mid$
$\quad (P(\alpha),F(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \longrightarrow \beta\ w) \mid (F(\alpha),P(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \longrightarrow \beta\ w) \mid$
$\quad \text{-} \Rightarrow ERR(dio)$

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

abbreviation $forall{::}({'a}{\Rightarrow}io\ opt){\Rightarrow}io\ opt\ (\forall\,)$ **where** $\forall\,\Phi \equiv case\ (\Phi\ da)\ of$
$\quad F(\varphi) \Rightarrow F(\lambda w.\forall\,x.\ case\ (\Phi\ x)\ of\ F(\psi) \Rightarrow \psi\ w)$
$\mid P(\varphi) \Rightarrow P(\lambda w.\forall\,x.\ case\ (\Phi\ x)\ of\ P(\psi) \Rightarrow \psi\ w) \mid \text{-} \Rightarrow ERR(dio)$
abbreviation $forallBinder{::}({'a}{\Rightarrow}io\ opt){\Rightarrow}io\ opt$ (**binder** $\forall$ $[8]\ 9$) **where** $\forall\,x.\ \varphi\ x \equiv \forall\,\varphi$

lemma *binderTest*: $(\forall\,x.\ \varphi\ x) = \forall\,(\lambda x.\ \varphi\ x)$ **by** *simp*

The modal $\Box$ 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 $\Box$ 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 $\Box$. $\Box\varphi$ is supported for formulas and propositional formulas.

abbreviation $box::io\ opt \Rightarrow io\ opt$ ($\Box$- [62] 63) **where** $\Box\varphi \equiv case\ \varphi\ of$
$F(\psi) \Rightarrow F(\lambda w.\forall v.\ \psi\ v) \mid P(\psi) \Rightarrow P(\lambda w.\forall v.\ \psi\ v) \mid \text{-} \Rightarrow ERR(dio)$

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 over 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 $(\lambda xy.\varphi)$, etc. Similar to before, we support only the cases where $n \leq 3$. Binder notation is introduced for $\boldsymbol{\lambda}$ (... unfortuntaley, I don't know yet how binder notation can be achieved also for $\boldsymbol{\lambda}^2$,$\boldsymbol{\lambda}^3$ ... need to find out.).

abbreviation $lam0::io\ opt \Rightarrow io\ opt$ ($\boldsymbol{\lambda}^0$) **where** $\boldsymbol{\lambda}^0\varphi \equiv case\ \varphi\ of$
$P(\psi) \Rightarrow P(\psi) \mid \text{-} \Rightarrow ERR\ dio$

abbreviation $lam::(e \Rightarrow io\ opt) \Rightarrow (e \Rightarrow io)\ opt$ ($\boldsymbol{\lambda}$) **where** $\boldsymbol{\lambda}\Phi \equiv case\ (\Phi\ de)\ of$
$P(\varphi) \Rightarrow T(\lambda x.\ case\ (\Phi\ x)\ of\ P(\varphi) \Rightarrow \varphi) \mid \text{-} \Rightarrow ERR(\lambda x.\ dio)$
abbreviation $lamBinder::(e \Rightarrow io\ opt) \Rightarrow (e \Rightarrow io)\ opt$ (**binder** $\boldsymbol{\lambda}$ [8] 9) **where** $\boldsymbol{\lambda}x.\ \varphi\ x \equiv \boldsymbol{\lambda}\ \varphi$

abbreviation $lam2::(e \Rightarrow e \Rightarrow io\ opt) \Rightarrow (e \Rightarrow e \Rightarrow io)\ opt$ ($\boldsymbol{\lambda}^2$) **where** $\boldsymbol{\lambda}^2\Phi \equiv case\ (\Phi\ de\ de)\ of$
$P(\varphi) \Rightarrow T(\lambda x\ y.\ case\ (\Phi\ x\ y)\ of\ P(\varphi) \Rightarrow \varphi) \mid \text{-} \Rightarrow ERR(\lambda x\ y.\ dio)$

abbreviation $lam3::(e \Rightarrow e \Rightarrow e \Rightarrow io\ opt) \Rightarrow (e \Rightarrow e \Rightarrow e \Rightarrow io)\ opt$ ($\boldsymbol{\lambda}^3$) **where** $\boldsymbol{\lambda}^3\Phi \equiv case\ (\Phi\ de\ de\ de)$ *of*
$P(\varphi) \Rightarrow T(\lambda x\ y\ z.\ case\ (\Phi\ x\ y\ z)\ of\ P(\varphi) \Rightarrow \varphi) \mid \text{-} \Rightarrow ERR(\lambda x\ y\ z.\ dio)$

The Principia Metaphysica supports rigid definite descriptions. Our definition maps $\boldsymbol{\iota}(\lambda x.\varphi)$ to $(THE\ x.\ \varphi\ cw)$, that is Isabelle's inbuilt definite description operator THE is utilized and evaluation is rigidly carried out with respect to the current world *cw*. We again introduce binder notation for $\boldsymbol{\iota}$.

abbreviation $that::(e \Rightarrow io\ opt) \Rightarrow e\ opt$ ($\boldsymbol{\iota}$) **where** $\boldsymbol{\iota}\Phi \equiv case\ (\Phi\ de)\ of$
$F(\varphi) \Rightarrow T(THE\ x.\ case\ (\Phi\ x)\ of\ F\ \psi \Rightarrow \psi\ dw) \mid P(\varphi) \Rightarrow T(THE\ x.\ case\ (\Phi\ x)\ of\ P\ \psi \Rightarrow \psi\ dw)$
$\mid \text{-} \Rightarrow ERR(de)$
abbreviation $thatBinder::(e \Rightarrow io\ opt) \Rightarrow e\ opt$ (**binder** $\boldsymbol{\iota}$ [8] 9) **where** $\boldsymbol{\iota}x.\ \varphi\ x \equiv \boldsymbol{\iota}\ \varphi$

lemma $(\!|F1^T,(\boldsymbol{\iota}x.\ \{\!|x^T,Q1^T|\!\}) |\!) = X$ **apply** *simp* **oops** — X is a propositional formula as intended

4 Further Logical Connectives

Further logical connectives can be defined as usual. For pragmatic reasons (to avoid the blow-up of abbreviation expansions) we prefer direct definitions in all cases.

abbreviation $conj::io\ opt \Rightarrow io\ opt \Rightarrow io\ opt$ (**infixl** $\wedge$ 53) **where** $\varphi \wedge \psi \equiv case\ (\varphi,\psi)\ of$
$(P(\alpha),P(\beta)) \Rightarrow P(\lambda w.\ \alpha\ w \wedge \beta\ w) \mid (F(\alpha),F(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \wedge \beta\ w) \mid$
$(P(\alpha),F(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \wedge \beta\ w) \mid (F(\alpha),P(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \wedge \beta\ w) \mid$
$\text{-} \Rightarrow ERR(dio)$

abbreviation $disj::io\ opt \Rightarrow io\ opt \Rightarrow io\ opt$ (**infixl** $\vee$ 52) **where** $\varphi \vee \psi \equiv case\ (\varphi,\psi)\ of$
$(P(\alpha),P(\beta)) \Rightarrow P(\lambda w.\ \alpha\ w \vee \beta\ w) \mid (F(\alpha),F(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \vee \beta\ w) \mid$
$(P(\alpha),F(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \vee \beta\ w) \mid (F(\alpha),P(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \vee \beta\ w) \mid$

- $\Rightarrow$ *ERR(dio)*

abbreviation *equiv*::*io opt*$\Rightarrow$*io opt*$\Rightarrow$*io opt* (**infixl** $\equiv$ *51*) **where** $\varphi \equiv \psi \equiv$ *case* (φ,ψ) *of*
$(P(\alpha),P(\beta)) \Rightarrow P(\lambda w.\ \alpha\ w \longleftrightarrow \beta\ w)$ | $(F(\alpha),F(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \longleftrightarrow \beta\ w)$ |
$(P(\alpha),F(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \longleftrightarrow \beta\ w)$ | $(F(\alpha),P(\beta)) \Rightarrow F(\lambda w.\ \alpha\ w \longleftrightarrow \beta\ w)$ |
- $\Rightarrow$ *ERR(dio)*

abbreviation *diamond*::*io opt*$\Rightarrow$*io opt* ($\Diamond$ - [*62*] *63*) **where** $\Diamond\varphi \equiv$ *case* φ *of*
$F(\psi) \Rightarrow F(\lambda w.\exists v.\ \psi\ v)$ | $P(\psi) \Rightarrow P(\lambda w.\exists v.\ \psi\ v)$ | - $\Rightarrow$ *ERR(dio)*

abbreviation *exists*::('*a*$\Rightarrow$*io opt*)$\Rightarrow$*io opt* ($\exists$) **where** $\exists\,\Phi \equiv$ *case* $(\Phi\ da)$ *of*
$P\ \varphi \Rightarrow P(\lambda w.\exists x.$ *case* $(\Phi\ x)$ *of* $P\ \psi \Rightarrow \psi\ w)$
| $F\ \varphi \Rightarrow F(\lambda w.\ \exists x.$ *case* $(\Phi\ x)$ *of* $F\ \psi \Rightarrow \psi\ w)$ | - $\Rightarrow$ *ERR dio*
abbreviation *existsBinder*::('*a*$\Rightarrow$*io opt*)$\Rightarrow$*io opt* (**binder** $\exists$ [*8*] *9*) **where** $\exists x.\ \varphi\ x \equiv \exists\varphi$

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 aro also noted as $\top$, $\bot$ and $*$. We could, of course, also introduce respective logical connectives for the meta-level, but in our applications (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). Actually, so far we only use validity.

abbreviation *valid* :: *io opt*$\Rightarrow$*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*$\Rightarrow$*mf* ($[\text{-}]^{sat}$ [*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*$\Rightarrow$*mf* ($[\text{-}]^{csat}$ [*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*$\Rightarrow$*mf* ($[\text{-}]^{inv}$ [*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

The next two statements are not theorems; Nitpick reports countermodels

lemma $[(\forall x.\ (\!|R^T,x^T|\!) \rightarrow \{\!|x^T,R^T|\!\})] = \top$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick
lemma $[(\forall x.\ \{\!|x^T,R^T|\!\} \rightarrow (\!|R^T,x^T|\!))] = \top$ **apply** *simp* **nitpick oops** — Countermodel by Nitpick

lemma $[(\forall y.\ (\!|R^T,y^T|\!))] = \top$ **apply** *simp* **nitpick oops**

However, the next two statements are of course valid.

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

6.1 Verifying Necessitation

The next two lemmata show that neccessitation 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 holds.

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. Our alternative, simpler definition of $\Box$ above omits this clause (since all worlds are reachable from any world in an equivalence relation). The following lemmata, which check various conditions for S5, confirm 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 the 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 construct is evaluated as ineligible at validity level; *error* $(*)$ is returned.

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).

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

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** $O! \equiv \boldsymbol{\lambda}x.\ \Diamond(\!|E^T,x^T|\!)$

lemma $O! = X$ **apply** *simp* **oops** — X is $(\lambda x\ w.\ Ex\ (exe1\ E\ x))^T$

Being abstract is is defined as not possibly being concrete.

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

lemma $A! = X$ **apply** *simp* **oops** — X is $(\lambda x\ w.\ \forall xa.\ \neg\ exe1\ E\ x\ xa)^T$

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

abbreviation *identityE*::e *opt*$\Rightarrow e$ *opt*$\Rightarrow io$ *opt* (**infixl** $=_E$ *63*) **where** $x =_E y \equiv$
$(\!|O!,x|\!) \wedge (\!|O!,y|\!) \wedge \Box(\forall F.\ (\!|F^T,x|\!) \equiv (\!|F^T,y|\!))$

lemma $a^T =_E a^T = X$ **apply** *simp* **oops** — X is "$(...)^P$

7.1 Identity on Individuals

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

lemma $a^T = a^T = X$ **apply** *simp* **oops** — X is $(...)^F$
lemma $((\!|A!,a^T|\!) \wedge (\!|A!,a^T|\!) \wedge \Box(\forall F.\ \{\!|a^T,F^T|\!\} \equiv \{\!|a^T,F^T|\!\})) = X$ **apply** *simp* **oops** — X is $(...)^F$
lemma $((\!|A!,a^T|\!) \wedge (\!|A!,a^T|\!)) = X$ **apply** *simp* **oops** — X is $(...)^P$
lemma $\Box(\forall F.\ \{\!|a^T,F^T|\!\} \equiv \{\!|a^T,F^T|\!\}) = X$ **apply** *simp* **oops** — X is $(...)^F$

As intended: the following two lambda-abstractions are not well-formed/eligible and their evaluation reports in ERR-terms.

lemma $\boldsymbol{\lambda}^2(\lambda x\ y.\ x^T = y^T) = X$ **apply** *simp* **oops** — X is $(\lambda x\ y.\ dio)^E$
lemma $(\boldsymbol{\lambda} x.\ x^T = y^T) = X$ **apply** *simp* **oops** — X is $(\lambda x.\ dio)^E$

7.2 Identitiy on Relations

abbreviation *identityRel1*:: $((e{\Rightarrow}io)\ opt){\Rightarrow}((e{\Rightarrow}io)\ opt){\Rightarrow}io\ opt$ (**infixl** $=^1$ *63*)
where $F1 =^1 G1 \equiv \Box(\forall x.\ \{\!|x^T,F1|\!\} \equiv \{\!|x^T,G1|\!\})$

abbreviation *identityRel2*:: $((e{\Rightarrow}e{\Rightarrow}io)\ opt){\Rightarrow}((e{\Rightarrow}e{\Rightarrow}io)\ opt){\Rightarrow}io\ opt$ (**infixl** $=^2$ *63*)
where $F2 =^2 G2 \equiv \forall x1.(\ (\boldsymbol{\lambda} y.(\!|F2,y^T,x1^T|\!)) =^1 (\boldsymbol{\lambda} y.(\!|G2,y^T,x1^T|\!))$
$\wedge (\boldsymbol{\lambda} y.(\!|F2,x1^T,y^T|\!)) =^1 (\boldsymbol{\lambda} y.(\!|G2,x1^T,y^T|\!)))$

abbreviation *identityRel3*:: $((e{\Rightarrow}e{\Rightarrow}e{\Rightarrow}io)\ opt){\Rightarrow}((e{\Rightarrow}e{\Rightarrow}e{\Rightarrow}io)\ opt){\Rightarrow}io\ opt$ (**infixl** $=^3$ *63*)
where $F3 =^3 G3 \equiv \forall x1\ x2.(\ (\boldsymbol{\lambda} y.(\!|F3,y^T,x1^T,x2^T|\!)) =^1 (\boldsymbol{\lambda} y.(\!|G3,y^T,x1^T,x2^T|\!))$
$\wedge (\boldsymbol{\lambda} y.(\!|F3,x1^T,y^T,x2^T|\!)) =^1 (\boldsymbol{\lambda} y.(\!|G3,x1^T,y^T,x2^T|\!))$
$\wedge (\boldsymbol{\lambda} y.(\!|F3,x1^T,x2^T,y^T|\!)) =^1 (\boldsymbol{\lambda} y.(\!|G3,x1^T,x2^T,y^T|\!)))$

lemma $F1^T =^1 G1^T = X$ **apply** *simp* **oops** — X is $(...)^F$
lemma $F2^T =^2 G2^T = X$ **apply** *simp* **oops** — X is $(...)^F$
lemma $F3^T =^3 G3^T = X$ **apply** *simp* **oops** — X is $(...)^F$
lemma $\{\!|x^T,F1^T|\!\} \equiv \{\!|x^T,G1^T|\!\} = X$ **apply** *simp* **oops** — X is $(...)^F$
lemma $(\!|F1^T,x^T|\!) \equiv (\!|G1^T,x^T|\!) = X$ **apply** *simp* **oops** — X is $(...)^P$
lemma $(\boldsymbol{\lambda} y.(\!|F2^T,y^T,x1^T|\!)){=}\ X$ **apply** *simp* **oops** — X is $(...)^T$

abbreviation *equalityRel0*::*io opt*⇒*io opt*⇒*io opt* (**infixl** $=^0$ *63*)
where $F0 =^0 G0 \equiv (\boldsymbol{\lambda} y\ .\ F0) =^1 (\boldsymbol{\lambda} y.\ G0)$

lemma $F1^T =^1 F1^T = X$ **apply** *simp* **oops** — X is $(...)^F$
lemma $[F1^T =^1 F1^T] = \top$ **apply** *simp* **done**
lemma $[F2^T =^2 F2^T] = \top$ **apply** *simp* **done**
lemma $[F3^T =^3 F3^T] = \top$ **apply** *simp* **done**

Some further tests:

We discuss the example from [7, pp.365-366]:

lemma $(\boldsymbol{\lambda} x.\ \exists F.\ \{\!|x^T,F^T|\!\} \rightarrow (\!|F^T,x^T|\!)) = X$ **apply** *simp* **oops** — X is $(\lambda x.\ dio)^E$

abbreviation K **where** $K \equiv \boldsymbol{\lambda} x.\exists F.(\{\!|x^T,F^T|\!\} \rightarrow (\!|F^T,x^T|\!))$

lemma $[(\exists x.\ (\!|A!,x^T|\!) \wedge (\forall F.\ (\{\!|x^T,F^T|\!\} \equiv F^T =^1 K)))] = *$ **apply** *simp* **done**
lemma $(\exists x.\ (\!|A!,x^T|\!) \wedge (\forall F.\ (\{\!|x^T,F^T|\!\} \equiv F^T =^1 K))) = X$ **apply** *simp* **oops** — X is $(dio)^E$

Tests on identity:

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* **oops**

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

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

7.3 Negation of Properties

abbreviation *notProp*::((*e*⇒*io*) *opt*)⇒(*e*⇒*io*) *opt* (∼- [*58*] *59*) **where** $\sim \Phi \equiv$ *case* Φ *of*
$T(\Psi) \Rightarrow \boldsymbol{\lambda} x.\neg(\!|\Phi,x^T|\!)$ | - ⇒ $ERR(deio)$

7.4 Individual Constant $\mathbf{a}_V$ and Function Term $\mathbf{a}_G$

abbreviation *a-V*::*e opt* ($\mathbf{a}_V$) **where** $\mathbf{a}_V \equiv \iota x.\ ((\!|A!,x^T|\!) \wedge (\forall F.\ \{\!|x^T,F^T|\!\} \equiv (F^T =^1 F^T)))$

abbreviation *a-G*::(*e*⇒*io*) *opt*⇒*e opt* ($\mathbf{a}$_ [*58*] *59*) **where** $\mathbf{a}_G \equiv \iota x.\ ((\!|A!,x^T|\!) \wedge (\forall F.\ \{\!|x^T,F^T|\!\} \equiv (F^T =^1 G)))$

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

8.5 Axioms of Necessity

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

lemma *a32-3-P*: $[\Box\Diamond\varphi^P \to \Diamond\varphi^P] = \top$ **apply** *simp* **done**
lemma *a32-3-F*: $[\Box\Diamond\varphi^F \to \Diamond\varphi^F] = \top$ **apply** *simp* **done**
lemma *a32-4-P*: $[(\forall\, x.\ \Box\varphi^P) \to \Box((\forall\, x.\ \varphi^P))] = \top$ **apply** *simp* **done**
lemma *a32-4-F*: $[(\forall\, x.\ \Box\varphi^F) \to \Box((\forall\, x.\ \varphi^F))] = \top$ **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**
lemma $\Diamond(\exists\, x.\ (\!|E^T, x^T|\!) \wedge \Diamond(\neg(\!|E^T, x^T|\!))) \wedge \Diamond(\neg(\exists\, x.\ (\!|E^T, x^T|\!) \wedge \Diamond(\neg(\!|E^T, x^T|\!)))) = X$ **apply** *simp* **oops** — X is $(...)^P$

8.6 Axioms of Necessity and Actuality

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

8.7 Axioms for Descriptions

lemma $(x^T = (\iota x.\{\!|x^T,R^T|\!\})) = X$ **apply** *simp* **oops** — X is $(...)^F$
lemma $(\forall\, z.\ (\boldsymbol{\mathcal{A}}(\{\!|x^T,R^T|\!\}) \equiv (z^T = x^T))) = X$ **apply** *simp* **oops** — X is $(...)^F$

For the following lemma cannot yet be automatically proved, since proof automation for definite descriptions is still not well enough developed in ATPs.

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

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.