

Modal Relational Type Theory in Isabelle/HOL

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

June 22, 2016

1 Introduction

The ambitious Principia Metaphysica [5] project at Stanford University aims at providing an encompassing axiomatic foundation for metaphysics, mathematics and the sciences. The starting point is Zalta’s theory of abstract objects [6], which

The theory of abstract objects utilizes a modal relational type theory (MRTT) as logical foundation. Arguments defending this choice against a modal functional type theory (MFTT) 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 extending Church’s theory of types [4], and recently it has been shown that Church’s type theory can be elegantly utilized as a meta-logic to semantically embed and automate various quantified non-classical logics, including MFTT [1, 2]. This embedding of MFTT has subsequently been employed in a case study in computational metaphysics, in which different variants of Kurt Gdel’s ontological argument were verified resp. falsified [2, 3].

The motivating research questions for the formalisation presented below include:

- Can functional type theory, despite the problems as pointed out by Zalta and Oppenheimer [7], nevertheless be fruitfully utilized to encode MRTT and subsequently the theory of abstract objects when adapting and utilizing the embeddings approach? As it turns, we will need a significant, technically involved extension of the embeddings approach.
- From another viewpoint the issue is to investigate possibilities to handle restricted forms of comprehension in a functional type theory context within the embedding approach.
- From a pragmatic point of view, we want to assess the elegance and user-friendliness of the proposed formalization? To what extent can Isabelle’s user interface be facilitated to hide unpleasant technicalities of the extended embedding from the user?
- Moreover, how far can automation be pushed in the approach to minimise user interaction in the formalization of the theory of abstract objects? For this note the proof automation has worked very well for the simpler embeddings of as used in related work [2, 3].

δ	$::= a_1, a_2, \dots$	δ	individual constants
ν	$::= x_1, x_2, \dots$	ν	individual variables
$(n \geq 0) \quad \Sigma^n$	$::= P_1^n, P_2^n, \dots$	Σ^n	n -place relation constants ($n \geq 0$)
$(n \geq 0) \quad \Omega^n$	$::= F_1^n, F_2^n, \dots$	Ω^n	n -place relation variables ($n \geq 0$)
α	$::= \nu \mid \Omega^n \ (n \geq 0)$	α	variables
κ	$::= \delta \mid \nu \mid \iota \nu \varphi$	κ	individual terms
$(n \geq 1) \quad \Pi^n$	$::= \Sigma^n \mid \Omega^n \mid [\lambda \nu_1 \dots \nu_n \varphi^*]$	Π^n	n -place relation terms ($n \geq 0$)
Π^0	$::= \Sigma^0 \mid \Omega^0 \mid [\lambda \varphi^*] \mid \varphi^*$	φ^*	propositional formulas
φ^*	$::= \Pi^n \kappa_1 \dots \kappa_n \ (n \geq 1) \mid \Pi^0 \mid (\neg \varphi^*) \mid (\varphi^* \rightarrow \varphi^*) \mid \forall \alpha \varphi^* \mid (\Box \varphi^*) \mid (\mathcal{A} \varphi^*)$	φ	formulas
φ	$::= \kappa_1 \Pi^1 \mid \varphi^* \mid (\neg \varphi) \mid (\varphi \rightarrow \varphi) \mid \forall \alpha \varphi \mid (\Box \varphi) \mid (\mathcal{A} \varphi)$	τ	terms
τ	$::= \kappa \mid \Pi^n \ (n \geq 0)$		

Figure 1: Grammar of MRTT, cf. [5] for further details. Two kinds of (complex) formulas are introduced: the φ -formulas may have encoding subformulas, while the φ^* -formulas must not. The latter are designated as propositional formulas, the former ones simply as formulas.

- Can the consistency of the theory eventually be validated with the available automated reasoning tools?
- Can the reasoners eventually even contribute some new knowledge?
- Are any suggestions arising for possible improvements in Isabelle/HOL. What are the particular problems detected in the course of the study?

In this contribution to the Archive of Formal Proofs, however, we primarily focus on presenting the encoding of MRTT in functional type theory. The idea is to extend the fundamental ideas underlying the previous encoding of MFTT in functional type theory. In subsequent papers we will then reuse and extend the foundations provided in this article and further address the above questions.

The encoding of modal functional type theory in functional type theory as explored in previous work is simple: modal logic formulas are identified with certain functional type theory formulas of predicate type $i \Rightarrow \text{bool}$ (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 for this work has been to suitably appropriately 'restrict' this embedding for modal relational type theory. The grammar of modal relational type theory is presented in Figure 1. Note that this grammar excludes terms such as $\lambda x. Rx \rightarrow xR$, where Rx represents the exemplification of property R by x and xR stands for the encoding of property R by x . The reason is that such kind of lambda-abstractions may lead to paradoxes in the theory of abstract objects [7].

To achieve our goal we provide means to explicitly represent, maintain and propagate information on the syntactical structure of MRTT in functional type theory. In particular, we

provide means in form of annotations to explicitly distinguish between propositional formulas, formulas, terms and erroneous (ineligible/excluded) formations. Respective annotation information is propagated from the innermost constituents to the top level constructions. This creates some non-trivial technical overhead. However, Isabelle/HOL’s user interface provide facilities to widely hide these technicalities from the user in applications.

A note on using abbreviations versus definitions in our approach: We are aware that abbreviations should be used sparsingly in Isabelle/HOL; they are automatically expanded and thus lead to a discrepancy between the internal and the external view of a term. However, we here deliberately deviate from this rule, since one aspect of the paper is to exactly illustrate this discrepancy and to emphasize the complexity of the embedding MRTT in functional type theory. In fact, this complexity makes pen and paper work with the proposed embedding pragmatically infeasible. In this sense, we agree with earlier findings [7]. On the other hand, we demonstrate the general feasibility here, and we show, that within a modern interactive proof assistant like Isabelle/HOL the approach can eventually be handled to some modest degree. In fact, as we will also illustrate, the simplifier *simp* of Isabelle/HOL is well capable of effectively reducing the technically inflated terms we obtain from the extended embedding again to their logical core content. In other words, the simplifier effectively analyses and rewrites the deeply annotated terms and propagates the annotation information as intended to top-level. It is exactly this effect which we want to illustrate and exploit here.¹

2 Preliminaries

We start out with some type declarations and type abbreviations. Note that 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 $(i \Rightarrow \text{bool})$. To make our representation more concise in the remainder we abbreviate this type as *io*.

typeddecl i
type-synonym $io = (i \Rightarrow \text{bool})$

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. The Theory of Abstract Objects later introduces means to distinguish between abstract and ordinary entities.

typeddecl e

To explicitly model the syntactical restrictions of MRTT we introduce a (polymorphic) datatype $'a \text{ opt}$ ($'a$ is a type variable) based on four constructors: *ERR* $'a$ (identifies ineligible/excluded constructions), *P* $'a$ (identifies propositional formulas), *F* $'a$ (identifies formulas), and *T* $'a$ (identifies eligible terms, such as lambda abstractions). The embedding approach (of MFTT in functional type theory) will be suitably adapted below so that for each language expression (in the embedded MRTT) the respective datatype is identified and appropriately propagated. The encapsulated expressions correspond to the previous embedding of MRTT in functional type theory [1, 2].

¹We have also experimented with using definitions instead of abbreviations; the respective encodings can be requested from the authors.

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

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

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

Certain language constructs in the Theory of Abstract objects, such as the actuality operator \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 the various other domains. We anyway assume that all domains are non-empty. Hence, introducing these constant symbols is obviously not harmful. ²

consts $dw::i$
consts $de::e \ dio::io \ deio::e \Rightarrow io \ da::'a$

3 Embedding of Modal Relational Type Theory

The various language constructs of MRTT (see Figure 1) are now introduced step by step.

The actuality operator \mathcal{A} , when being applied to a formula or propositional formula φ , evaluates φ wrt the fixed given world dw . The compound expression $\mathcal{A}\varphi$ inherits its syntactical category F (formula) or P (propositional formula) from φ . If the syntactical category of φ is ERR (error) or T (term), then the syntactical category of $\mathcal{A}\varphi$ is ERR and a dummy entity of appropriate type is returned. This illustrates the core ideas of our explicit modeling of MRTT grammatical structure in functional type theory. This scheme will be repeated below for all the other language constructs of MRTT.

abbreviation $\text{Actual}::io \text{ opt } \Rightarrow io \text{ opt } (\mathcal{A} \ - \ [64] \ 65)$ **where** $\mathcal{A}\varphi \equiv \text{case } \varphi \text{ of}$
 $F(\psi) \Rightarrow F(\lambda w. \ \psi \ dw) \mid P(\psi) \Rightarrow P(\lambda w. \ \psi \ dw) \mid - \Rightarrow \text{ERR}(dio)$

The Theory of Abstract Objects distinguishes between encoding properties $\kappa_1\Pi^1$ and exemplifying properties $\Pi^n, \kappa_1, \dots, \kappa_n$ (for $n \geq 1$).

Encoding $\kappa_1\Pi^1$ is noted below as $\llbracket \kappa_1, \Pi^1 \rrbracket$. Encoding yields formulas and never propositional formulas. It is mapped to expressions of form $(\text{enc } Q \ y)$, where enc is uninterpreted constant symbol of appropriate type. Exemplification will instead be mapped to predicate application. Unfortunately, we cannot map both, exemplification and encoding, to predicate application, since this would have undesirable effects. For example, we could then prove that $[(\forall x. \llbracket R^T, x^T \rrbracket) \equiv \llbracket R^T, x^T \rrbracket]$ is true, which we definitely want to avoid. See also the basic tests in Subsection 6.1.

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

²The single polymorphic dummy $da::'a$, utilized e.g. in the definition of the universal quantifier of MRTT below, actually covers already all cases. However, to avoid unnecessary type inferences we actually prefer non-polymorphic dummy elements in all those cases where we can statically determine the required type.

Unary exemplifying formulas $\Pi^1 \kappa_1$ are noted below as $\langle \Pi^1, \kappa_1 \rangle$. Exemplification yields propositional formulas. Like encoding, it is then mapped to predicate application.

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

For pragmatcal reasons we support exemplification formulas $\Pi^n, \kappa_1, \dots, \kappa_n$ here only for $1 \leq n \leq 3$. In addition to the unary case above, we thus introduce two further cases.

abbreviation *Exe2*::($e \Rightarrow e \Rightarrow io$) $opt \Rightarrow e$ $opt \Rightarrow e$ $opt \Rightarrow io$ opt ($\langle -, -, - \rangle$)

where $\langle \Phi, x1, x2 \rangle \equiv \text{case } (\Phi, x1, x2) \text{ of}$
 $(T(Q), T(y1), T(y2)) \Rightarrow P(Q \ y1 \ y2) \mid - \Rightarrow \text{ERR}(\text{dio})$

abbreviation *Exe3*::($e \Rightarrow e \Rightarrow e \Rightarrow io$) $opt \Rightarrow e$ $opt \Rightarrow e$ $opt \Rightarrow e$ $opt \Rightarrow io$ opt ($\langle -, -, -, - \rangle$)

where $\langle \Phi, x1, x2, x3 \rangle \equiv \text{case } (\Phi, x1, x2, x3) \text{ of}$
 $(T(Q), T(y1), T(y2), T(y3)) \Rightarrow P(Q \ y1 \ y2 \ y3) \mid - \Rightarrow \text{ERR}(\text{dio})$

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 if both constituents are propositional formulas. If at least one constituent 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 - [58] 59) **where** $\neg \varphi \equiv \text{case } \varphi \text{ of}$

$F(\psi) \Rightarrow F(\lambda w. \neg(\psi \ w)) \mid P(\psi) \Rightarrow P(\lambda w. \neg(\psi \ w)) \mid - \Rightarrow \text{ERR}(\text{dio})$

abbreviation *implies*:: $io \ opt \Rightarrow io \ opt \Rightarrow io \ opt$ (**infixl** \rightarrow 51) **where** $\varphi \rightarrow \psi \equiv \text{case } (\varphi, \psi) \text{ of}$

$(P(\alpha), P(\beta)) \Rightarrow P(\lambda w. \alpha \ w \longrightarrow \beta \ w) \mid (F(\alpha), F(\beta)) \Rightarrow F(\lambda w. \alpha \ w \longrightarrow \beta \ w) \mid$
 $(P(\alpha), F(\beta)) \Rightarrow F(\lambda w. \alpha \ w \longrightarrow \beta \ w) \mid (F(\alpha), P(\beta)) \Rightarrow F(\lambda w. \alpha \ w \longrightarrow \beta \ w) \mid$
 $- \Rightarrow \text{ERR}(\text{dio})$

Also universal quantification $\forall (\lambda x. \varphi)$ (first-order and higher-order) is supported for both, formulas and propositional formulas. Following previous work, the embedding maps $\forall (\lambda x. \varphi)$ to $(\lambda w. \forall x. \varphi \ w)$, where the latter \forall is the universal quantifier from the HOL meta-logic. Note that \forall is introduced as logical connective based on the existing λ -binder. To improve the presentation and intuitive use in the remainder we additionally 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** $\forall \Phi \equiv \text{case } (\Phi \ da) \text{ of}$

$F(-) \Rightarrow F(\lambda w. \forall x. \text{case } (\Phi \ x) \text{ of } F(\psi) \Rightarrow \psi \ w)$
 $\mid P(-) \Rightarrow P(\lambda w. \forall x. \text{case } (\Phi \ x) \text{ of } P(\psi) \Rightarrow \psi \ w) \mid - \Rightarrow \text{ERR}(\text{dio})$

abbreviation *forallBinder*::($'a \Rightarrow io \ opt \Rightarrow io \ opt$) (**binder** \forall [8] 9) **where** $\forall x. \varphi \equiv \forall \varphi$

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 also improves the efficiency of theorem provers, cf. [3]. In Section 6.4 we will actually demonstrate that the expected S5 properties are validated by our modeling of \Box . The \Box -operator can be applied to formulas and propositional formulas.

abbreviation *box*:: $io \ opt \Rightarrow io \ opt$ (\Box - [62] 63) **where** $\Box \varphi \equiv \text{case } \varphi \text{ of}$

$F(\psi) \Rightarrow F(\lambda w. \forall v. \psi \ v) \mid P(\psi) \Rightarrow P(\lambda w. \forall v. \psi \ v) \mid - \Rightarrow \text{ERR}(\text{dio})$

n-ary lambda abstraction $\lambda^0, \lambda, \lambda^2, \lambda^3, \dots$, for $n \geq 0$, is supported in the theory of abstract objects only for propositional formulas. This way constructs such as beforehand mentioned $(\lambda x. Rx \rightarrow xR)$ (noted here as $(\lambda x. \langle R^T, x^T \rangle \rightarrow \langle x^T, R^T \rangle)$) are excluded, respectively identified

as *ERR*-annotated terms in our framework. Their embedding is straightforward: λ^0 is mapped to identity and $\lambda, \lambda^2, \lambda^3, \dots$ are mapped to n -ary lambda abstractions, that is, $\lambda(\lambda x. \varphi)$ is mapped to $(\lambda x. \varphi)$ and $\lambda^2(\lambda xy. \varphi)$ to $(\lambda xy. \varphi)$, etc. Similar to before, we support only the cases for $n \leq 3$. Binder notation is introduced for λ .³

abbreviation $lam0::io\ opt \Rightarrow io\ opt\ (\lambda^0)$ **where** $\lambda^0 \varphi \equiv case\ \varphi\ of$

$P(\psi) \Rightarrow P(\psi) \mid - \Rightarrow ERR\ dio$

abbreviation $lam::(e \Rightarrow io\ opt) \Rightarrow (e \Rightarrow io)\ opt\ (\lambda)$ **where** $\lambda \Phi \equiv case\ (\Phi\ de)\ of$

$P(-) \Rightarrow T(\lambda x. case\ (\Phi\ x)\ of\ P(\varphi) \Rightarrow \varphi) \mid - \Rightarrow ERR(\lambda x. dio)$

abbreviation $lamBinder::(e \Rightarrow io\ opt) \Rightarrow (e \Rightarrow io)\ opt\ (\text{binder } \lambda\ [8]\ 9)$ **where** $\lambda x. \varphi \equiv \lambda\ \varphi$

abbreviation $lam2::(e \Rightarrow e \Rightarrow io\ opt) \Rightarrow (e \Rightarrow e \Rightarrow io)\ opt\ (\lambda^2)$ **where** $\lambda^2 \Phi \equiv case\ (\Phi\ de\ de)\ of$

$P(-) \Rightarrow T(\lambda x\ y. case\ (\Phi\ x\ y)\ of\ P(\varphi) \Rightarrow \varphi) \mid - \Rightarrow ERR(\lambda x\ y. dio)$

abbreviation $lam3::(e \Rightarrow e \Rightarrow e \Rightarrow io\ opt) \Rightarrow (e \Rightarrow e \Rightarrow e \Rightarrow io)\ opt\ (\lambda^3)$ **where** $\lambda^3 \Phi \equiv case\ (\Phi\ de\ de\ de)\ of$

$P(-) \Rightarrow T(\lambda x\ y\ z. case\ (\Phi\ x\ y\ z)\ of\ P(\varphi) \Rightarrow \varphi) \mid - \Rightarrow ERR(\lambda x\ y\ z. dio)$

The theory of abstract objects supports rigid definite descriptions. Our definition maps $\iota(\lambda x. \varphi)$ to $(THE\ x. \varphi\ dw)$, that is, Isabelle's inbuilt definite description operator *THE* is utilized and evaluation is rigidly carried out with respect to the current world denoted by *dw*. We again introduce binder notation for ι .

abbreviation $that::(e \Rightarrow io\ opt) \Rightarrow e\ opt\ (\iota)$ **where** $\iota \Phi \equiv case\ (\Phi\ de)\ of$

$F(-) \Rightarrow T(THE\ x. case\ (\Phi\ x)\ of\ F\ \psi \Rightarrow \psi\ dw) \mid P(-) \Rightarrow T(THE\ x. case\ (\Phi\ x)\ of\ P\ \psi \Rightarrow \psi\ dw) \mid - \Rightarrow ERR(de)$

abbreviation $thatBinder::(e \Rightarrow io\ opt) \Rightarrow e\ opt\ (\text{binder } \iota\ [8]\ 9)$ **where** $\iota x. \varphi \equiv \iota\ \varphi$

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 $conj::io\ opt \Rightarrow io\ opt \Rightarrow io\ opt\ (\text{infixl } \wedge\ 53)$ **where** $\varphi \wedge \psi \equiv case\ (\varphi, \psi)\ of$

$(P(\alpha), P(\beta)) \Rightarrow P(\lambda w. \alpha\ w \wedge \beta\ w) \mid (F(\alpha), F(\beta)) \Rightarrow F(\lambda w. \alpha\ w \wedge \beta\ w) \mid$

$(P(\alpha), F(\beta)) \Rightarrow F(\lambda w. \alpha\ w \wedge \beta\ w) \mid (F(\alpha), P(\beta)) \Rightarrow F(\lambda w. \alpha\ w \wedge \beta\ w) \mid$

$- \Rightarrow ERR(dio)$

abbreviation $disj::io\ opt \Rightarrow io\ opt \Rightarrow io\ opt\ (\text{infixl } \vee\ 52)$ **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 $equiv::io\ opt \Rightarrow io\ opt \Rightarrow io\ opt\ (\text{infixl } \equiv\ 51)$ **where** $\varphi \equiv \psi \equiv case\ (\varphi, \psi)\ of$

$(P(\alpha), P(\beta)) \Rightarrow P(\lambda w. \alpha\ w \longleftrightarrow \beta\ w) \mid (F(\alpha), F(\beta)) \Rightarrow F(\lambda w. \alpha\ w \longleftrightarrow \beta\ w) \mid$

$(P(\alpha), F(\beta)) \Rightarrow F(\lambda w. \alpha\ w \longleftrightarrow \beta\ w) \mid (F(\alpha), P(\beta)) \Rightarrow F(\lambda w. \alpha\ w \longleftrightarrow \beta\ w) \mid$

$- \Rightarrow ERR(dio)$

abbreviation $diamond::io\ opt \Rightarrow io\ opt\ (\Diamond -\ [62]\ 63)$ **where** $\Diamond \varphi \equiv case\ \varphi\ of$

$F(\psi) \Rightarrow F(\lambda w. \exists v. \psi\ v) \mid P(\psi) \Rightarrow P(\lambda w. \exists v. \psi\ v) \mid - \Rightarrow ERR(dio)$

abbreviation $exists::('a \Rightarrow io\ opt) \Rightarrow io\ opt\ (\exists)$ **where** $\exists \Phi \equiv case\ (\Phi\ da)\ of$

³Unfortunately, we could not find out how suitable binder notation could be analogously provided for λ^2 and λ^3

$P(-) \Rightarrow P(\lambda w. \exists x. \text{case } (\Phi x) \text{ of } P \psi \Rightarrow \psi w)$
 $| F(-) \Rightarrow F(\lambda w. \exists x. \text{case } (\Phi x) \text{ of } F \psi \Rightarrow \psi w) | - \Rightarrow \text{ERR dio}$
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 are also noted as \top , \perp 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* (\perp) | *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: $[\varphi]$ (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 ([\cdot] [1]) **where** $[\varphi] \equiv \text{case } \varphi \text{ of}$

$P(\psi) \Rightarrow \text{if } \forall w. (\psi w) \longleftrightarrow \text{True then } \top \text{ else } \perp$

$| F(\psi) \Rightarrow \text{if } \forall w. (\psi w) \longleftrightarrow \text{True then } \top \text{ else } \perp | - \Rightarrow *$

abbreviation *satisfiable* :: io opt \Rightarrow mf ($[\cdot]^{sat}$ [1]) **where** $[\varphi]^{sat} \equiv \text{case } \varphi \text{ of}$

$P(\psi) \Rightarrow \text{if } \exists w. (\psi w) \longleftrightarrow \text{True then } \top \text{ else } \perp$

$| F(\psi) \Rightarrow \text{if } \exists w. (\psi w) \longleftrightarrow \text{True then } \top \text{ else } \perp | - \Rightarrow *$

abbreviation *countersatisfiable* :: io opt \Rightarrow mf ($[\cdot]^{csat}$ [1]) **where** $[\varphi]^{csat} \equiv \text{case } \varphi \text{ of}$

$P(\psi) \Rightarrow \text{if } \exists w. \neg(\psi w) \longleftrightarrow \text{True then } \top \text{ else } \perp$

$| F(\psi) \Rightarrow \text{if } \exists w. \neg(\psi w) \longleftrightarrow \text{True then } \top \text{ else } \perp | - \Rightarrow *$

abbreviation *invalid* :: io opt \Rightarrow mf ($[\cdot]^{inv}$ [1]) **where** $[\varphi]^{inv} \equiv \text{case } \varphi \text{ of}$

$P(\psi) \Rightarrow \text{if } \forall w. \neg(\psi w) \longleftrightarrow \text{True then } \top \text{ else } \perp$

$| F(\psi) \Rightarrow \text{if } \forall w. \neg(\psi w) \longleftrightarrow \text{True then } \top \text{ else } \perp | - \Rightarrow *$

6 Some Basic Tests

6.1 Exemplification and Encoding

For the following non-theorems we indeed get countermodels by nitpick.

lemma $[(\forall x. (\langle R^T, x^T \rangle \rightarrow \langle x^T, R^T \rangle)) = \top]$ **apply simp nitpick** [*expect = genuine*] **oops** — Countermodel by Nitpick

lemma $[(\forall x. \langle x^T, R^T \rangle \rightarrow (\langle R^T, x^T \rangle)) = \top]$ **apply simp nitpick** [*expect = genuine*] **oops** — Countermodel by Nitpick

With this example we also want to illustrate the inflation of representations as caused by the embedding. For this note, that the formula $[(\forall x. (\langle R^T, x^T \rangle \rightarrow \langle x^T, R^T \rangle)) = \top]$ abbreviates the actual term $(\text{case case } \langle da^T, R^T \rangle \rightarrow (\langle R^T, da^T \rangle) \text{ of } P x \Rightarrow (\lambda w. \forall x. \text{case } \langle x^T, R^T \rangle \rightarrow (\langle R^T, x^T \rangle) \text{ of } P \psi \Rightarrow \psi w)^P | F x \Rightarrow (\lambda w. \forall x. \text{case } \langle x^T, R^T \rangle \rightarrow (\langle R^T, x^T \rangle) \text{ of } F \psi \Rightarrow \psi w)^F | - \Rightarrow \text{dio}^E \text{ of } P \psi \Rightarrow \text{if } \forall w. \psi w = \text{True then } \top \text{ else } \perp | F \psi \Rightarrow \text{if } \forall w. \psi w = \text{True then } \top \text{ else } \perp | - \Rightarrow *) = \top$. In Isabelle the inflated term is displayed in the output window when placing the mouse on the abbreviated representation. However, the simplifier is capable

of evaluating the annotations and thereby reducing this inflated term again to $\forall w x. R x w \rightarrow enc R x w$ as intended; one can easily see this when placing the mouse on "simp". Below we will see that the inflated representations can easily fill several pages for abbreviated formulas which are only slightly longer than what we have here. This provides evidence for the pragmatic infeasibility of the approach when using pen and paper only.

The next two statements are valid and the simplifier quickly proves this.

lemma $[(\forall x. (\llbracket R^T, x^T \rrbracket \rightarrow \llbracket R^T, x^T \rrbracket)) = \top]$ **by simp**

lemma $[(\forall x. (\llbracket x^T, R^T \rrbracket \rightarrow \llbracket x^T, R^T \rrbracket)) = \top]$ **by simp**

6.2 Verifying Necessitation

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

lemma *necessitationF*: $[\varphi^F] = \top \rightarrow [\Box \varphi^F] = \top$ **by simp**

lemma *necessitationP*: $[\varphi^P] = \top \rightarrow [\Box \varphi^P] = \top$ **by simp**

6.3 Modal Collapse is Countersatisfiable

The modelfinder Nitpick constructs a finite countermodel to the assertion that modal collapse holds.

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

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

6.4 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**

6.5 Relations between Meta-Logical Notions

lemma $[\varphi^P] = \top \iff [\varphi^P]^{csat} = \perp$ **apply simp done**

lemma $[\varphi^P]^{sat} = \top \iff [\varphi^P]^{inv} = \perp$ **apply simp done**

lemma $[\varphi^F] = \top \iff [\varphi^F]^{csat} = \perp$ **apply simp done**

lemma $[\varphi^F]^{sat} = \top \iff [\varphi^F]^{inv} = \perp$ **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**

6.6 Testing the Propagation of Syntactical Category Information

lemma $\exists X. (\llbracket R^T, a^T \rrbracket = X^P \wedge \neg(\exists X. (\llbracket R^T, a^T \rrbracket = X^F) \wedge \neg(\exists X. (\llbracket R^T, a^T \rrbracket = X^T) \wedge \neg(\exists X. (\llbracket R^T, a^T \rrbracket = X^E))$ **apply simp done**

lemma $\exists X. \llbracket x^T, R^T \rrbracket = X^F \wedge \neg(\exists X. \llbracket x^T, R^T \rrbracket = X^P) \wedge \neg(\exists X. \llbracket x^T, R^T \rrbracket = X^T) \wedge \neg(\exists X. \llbracket x^T, R^T \rrbracket = X^E)$ **apply simp done**

Most importantly, we have that the following language construct is evaluated as ineligible at validity level; *error* (*) is returned.

lemma $(\lambda x. (\llbracket R^T, x^T \rrbracket \rightarrow \llbracket x^T, R^T \rrbracket)) = X$ **apply simp oops**

lemma $(\llbracket \lambda x. (\llbracket R^T, x^T \rrbracket \rightarrow \llbracket x^T, R^T \rrbracket, a^T) \rrbracket = *$ **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 $(\llbracket \lambda x. (\llbracket R^T, x^T \rrbracket \rightarrow \llbracket x^T, R^T \rrbracket, a^T) \rrbracket = X$ **apply simp oops** — X is dio^E

lemma $(\llbracket \lambda x. (\llbracket R^T, x^T \rrbracket \wedge \neg \llbracket x^T, R^T \rrbracket, a^T) \rrbracket = X$ **apply simp oops** — X is dio^E

6.7 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 $ordinaryObject::(e \Rightarrow io) \ opt \ (O!) \ \text{where} \ O! \equiv \lambda x. \Diamond(\llbracket E^T, x^T \rrbracket)$

lemma $O! = X \ \text{apply} \ simp \ \text{oops} \quad \text{---} \ X \text{ is } (\lambda x \ w. \ Ex \ (exe \ E \ x))^T$

Being abstract is defined as not possibly being concrete.

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

lemma $A! = X \ \text{apply} \ simp \ \text{oops} \quad \text{---} \ X \text{ is } (\lambda x \ w. \ \forall xa. \neg \ exe \ (E \ x) \ xa)^T$

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

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

lemma $a^T =_E a^T = X \ \text{apply} \ simp \ \text{oops} \quad \text{---} \ X \text{ is } "(...)^P$

7.0.1 Remark: Nested lambda-expressions

lemma $(\lambda x. x^T =_E a^T) = X \ \text{apply} \ simp \ \text{oops}$

lemma $(\lambda x. x^T =_E a^T) = (\lambda x. a^T =_E x^T) \ \text{apply} \ simp \ \text{by} \ \text{metis}$

7.1 Identity on Individuals

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

7.1.1 Remark: Tracing the propagation of annotations

lemma $a^T = a^T = X \ \text{apply} \ simp \ \text{oops}$

--- X is $(...)^F$

lemma $(\llbracket A!, a^T \rrbracket \wedge \llbracket A!, a^T \rrbracket \wedge \Box(\forall F. \llbracket a^T, F^T \rrbracket \equiv \llbracket a^T, F^T \rrbracket)) = X \ \text{apply} \ simp \ \text{oops} \quad \text{---} \ X \text{ is } (...)^F$

lemma $(\llbracket A!, a^T \rrbracket \wedge \llbracket A!, a^T \rrbracket) = X \ \text{apply} \ simp \ \text{oops}$

--- X is $(...)^P$

lemma $\Box(\forall F. \llbracket a^T, F^T \rrbracket \equiv \llbracket a^T, F^T \rrbracket) = X \ \text{apply} \ simp \ \text{oops}$

--- X is $(...)^F$

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

lemma $\lambda^2(\lambda x \ y. x^T = y^T) = X \ \text{apply} \ simp \ \text{oops} \quad \text{---} \ X \text{ is } (\lambda x \ y. \text{dio})^E$

lemma $(\lambda x. x^T = y^T) = X \ \text{apply} \ simp \ \text{oops} \quad \text{---} \ X \text{ is } (\lambda x. \text{dio})^E$

7.2 Identitiy on Relations

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

abbreviation $identityRel2::((e \Rightarrow e \Rightarrow io) \ opt) \Rightarrow ((e \Rightarrow e \Rightarrow io) \ opt) \Rightarrow io \ opt \ (\text{infixl} \ =^2 \ 63)$

where $F2 =^2 G2 \equiv \forall x1. (\lambda y. \langle F2, y^T, x1^T \rangle) =^1 (\lambda y. \langle G2, y^T, x1^T \rangle)$
 $\wedge (\lambda y. \langle F2, x1^T, y^T \rangle) =^1 (\lambda y. \langle G2, x1^T, y^T \rangle)$

abbreviation *identityRel3*:: $((e \Rightarrow e \Rightarrow e \Rightarrow io) \text{ opt}) \Rightarrow ((e \Rightarrow e \Rightarrow e \Rightarrow io) \text{ opt}) \Rightarrow io \text{ opt}$ (**infixl** =³ 63)

where $F3 =^3 G3 \equiv \forall x1 \ x2. (\lambda y. \langle F3, y^T, x1^T, x2^T \rangle) =^1 (\lambda y. \langle G3, y^T, x1^T, x2^T \rangle)$
 $\wedge (\lambda y. \langle F3, x1^T, y^T, x2^T \rangle) =^1 (\lambda y. \langle G3, x1^T, y^T, x2^T \rangle)$
 $\wedge (\lambda y. \langle F3, x1^T, x2^T, y^T \rangle) =^1 (\lambda y. \langle G3, x1^T, x2^T, y^T \rangle)$

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 $\langle x^T, F1^T \rangle \equiv \langle x^T, G1^T \rangle = X$ **apply simp oops** — X is (...)^F

lemma $\langle F1^T, x^T \rangle \equiv \langle G1^T, x^T \rangle = X$ **apply simp oops** — X is (...)^P

lemma $(\lambda y. \langle F2^T, y^T, x1^T \rangle) = X$ **apply simp oops** — X is (...)^T

abbreviation *equalityRel0*:: $io \text{ opt} \Rightarrow io \text{ opt} \Rightarrow io \text{ opt}$ (**infixl** =⁰ 63)

where $F0 =^0 G0 \equiv (\lambda y. F0) =^1 (\lambda y. G0)$

Some tests: reflexivity, symmetry, transitivity

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**

lemma $[(F1^T =^1 G1^T) \equiv (G1^T =^1 F1^T)] = \top$ **apply simp by auto**

lemma $[(F2^T =^2 G2^T) \equiv (G2^T =^2 F2^T)] = \top$ **apply simp by auto**

lemma $[(F3^T =^3 G3^T) \equiv (G3^T =^3 F3^T)] = \top$ **apply simp by auto**

lemma $[(F1^T =^1 G1^T) \wedge (G1^T =^1 H1^T) \rightarrow (F1^T =^1 H1^T)] = \top$ **by simp**

lemma $[(F2^T =^2 G2^T) \wedge (G2^T =^2 H2^T) \rightarrow (F2^T =^2 H2^T)] = \top$ **by simp**

lemma $[(F3^T =^3 G3^T) \wedge (G3^T =^3 H3^T) \rightarrow (F3^T =^3 H3^T)] = \top$ **by simp**

The above examples are very resource intensive already

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

lemma $(\lambda x. \exists F. \langle x^T, F^T \rangle \rightarrow \langle F^T, x^T \rangle) = X$ **apply simp oops** — X is $(\lambda x. dio)^E$

abbreviation *K* where $K \equiv \lambda x. \exists F. (\langle x^T, F^T \rangle \rightarrow \langle F^T, x^T \rangle)$

lemma $K = X$ **apply simp oops** — X is $(\lambda x. dio)^E$

lemma $[(\exists x. \langle A!, x^T \rangle) \wedge (\forall F. (\langle x^T, F^T \rangle \equiv F^T =^1 K)))] = *$ **apply simp done**

lemma $(\exists x. \langle A!, x^T \rangle) \wedge (\forall F. (\langle x^T, F^T \rangle \equiv F^T =^1 K)) = X$ **apply simp oops** — X is $(dio)^E$

Tests on identity:

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

lemma $[\langle O!, a^T \rangle \rightarrow a^T =_E a^T] = \top$ **apply simp done**

lemma $[(\forall F. \langle F^T, x^T \rangle \equiv \langle F^T, x^T \rangle)] = \top$ **apply simp done**

lemma $[\langle O!, a^T \rangle \rightarrow \langle \lambda x. x^T =_E a^T, a^T \rangle] = \top$ **apply simp done**

lemma $[(a^T =_E a^T) \equiv \langle \lambda x. x^T =_E a^T, a^T \rangle] = \top$ **apply simp done**

lemma $[(a^T =_E a^T) \equiv \llbracket a^T, \lambda x. x^T =_E a^T \rrbracket] = \top$ **apply simp nitpick oops** — Countermodel by nitpick, because of "enc"

lemma $[(\exists F. \llbracket a^T, F^T \rrbracket)] = \top$ **apply simp nitpick oops** — Countermodel by Nitpick

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 $notProp::(e \Rightarrow io) \Rightarrow (e \Rightarrow io) \Rightarrow (\sim - [58] 59)$ **where** $\sim \Phi \equiv case \Phi of$
 $T(\Psi) \Rightarrow \lambda x. \neg \llbracket \Phi, x^T \rrbracket \mid - \Rightarrow ERR(deio)$

7.4 Individual Constant a_V and Function Term a_G

abbreviation $a-V::e \Rightarrow (a_V)$ **where** $a_V \equiv \iota x. (\llbracket A!, x^T \rrbracket \wedge (\forall F. \llbracket x^T, F^T \rrbracket \equiv (F^T =^1 F^T)))$

abbreviation $a-G::(e \Rightarrow io) \Rightarrow e \Rightarrow (a- [58] 59)$ **where** $a_G \equiv \iota x. (\llbracket A!, x^T \rrbracket \wedge (\forall F. \llbracket x^T, F^T \rrbracket \equiv (F^T =^1 G)))$

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

Here I have a big problem

lemma $a31-1-P: [\mathcal{A}\varphi^P \equiv \varphi^P] = \top$ **apply simp nitpick oops**

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: $[\Diamond(\exists x. (\Diamond E^T, x^T) \wedge \Diamond(\neg(\Diamond E^T, x^T))) \wedge \Diamond(\neg(\exists x. (\Diamond E^T, x^T) \wedge \Diamond(\neg(\Diamond E^T, x^T))))] = \top$

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

lemma $[\Diamond(\exists x. (\Diamond E^T, x^T) \wedge \Diamond(\neg(\Diamond E^T, x^T))) \wedge \Diamond(\neg(\exists x. (\Diamond E^T, x^T) \wedge \Diamond(\neg(\Diamond E^T, x^T))))] \neq *$ **apply simp done**
lemma $\Diamond(\exists x. (\Diamond E^T, x^T) \wedge \Diamond(\neg(\Diamond E^T, x^T))) \wedge \Diamond(\neg(\exists x. (\Diamond E^T, x^T) \wedge \Diamond(\neg(\Diamond E^T, x^T)))) = X$ **apply simp oops** — X is $(\dots)^P$

8.6 (Instances of) Barcan Formula and Converse Barcan Formula

lemma *BF-inst*: $[(\forall \alpha. \Box(\Diamond R^T, \alpha^T)) \rightarrow \Box(\forall \alpha. (\Diamond R^T, \alpha^T))] = \top$ **by simp**
lemma *CBF-inst*: $[\Box(\forall \alpha. (\Diamond R^T, \alpha^T)) \rightarrow (\forall \alpha. \Box(\Diamond R^T, \alpha^T))] = \top$ **apply simp by auto**

8.7 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**

8.8 Axioms for Descriptions

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

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*: $[(x^T = (\iota x. (\Diamond x^T, R^T))) \equiv (\forall z. (\mathcal{A}(\Diamond z^T, R^T) \equiv (z^T = x^T)))] = \top$ **apply simp oops**

8.9 Axioms for Complex relation Terms

We check for some α -renaming instances

lemma $(\lambda z. (\Diamond R^T, z^T, (\iota y. (\Diamond Q^T, y^T)))) = (\lambda x. (\Diamond R^T, x^T, (\iota z. (\Diamond Q^T, z^T))))$ **apply simp done**
lemma $((\forall F. (\Diamond F^T, a^T)) \equiv (\forall G. (\Diamond G^T, b^T))) = (\forall F. (\Diamond F^T, a^T)) \equiv (\forall F. (\Diamond F^T, b^T))$ **apply simp done**

Others are analogously valid, we omit them here

8.10 Axioms of Encoding

The following need to become an axioms; they are not implied by the embedding.

axiomatization where

a36: $\llbracket \{x^T, G^T\} \rightarrow \Box \{x^T, G^T\} \rrbracket = \top$ **and**
a37: $\llbracket \mathcal{A} \{x^T, G^T\} \rightarrow \{x^T, G^T\} \rrbracket = \top$

The following however holds

lemma $\llbracket \Box (\mathcal{A} \{x^T, G^T\} \rightarrow \{x^T, G^T\}) \rrbracket = \top$ **apply simp nitpick oops**

9 Leibniz Theory of Concepts

Below we don't get that far yet, a systematic bottom up development seems to be required first

abbreviation *LeibnizianConcept* :: $(e \Rightarrow io) \text{ opt } (C!)$
where $C! \equiv \lambda x. \langle A!, x^T \rangle$
abbreviation *ConceptSummation* (**infix** \oplus 100)
where $x \oplus y \equiv \iota z. (\langle C!, x \rangle \wedge (\forall F. (\llbracket z^T, F^T \rrbracket \equiv \llbracket x, F^T \rrbracket \vee \llbracket y, F^T \rrbracket)))$
abbreviation *ConceptInclusion* (**infix** \preceq 100)
where $x \preceq y \equiv \forall F. (\llbracket x, F^T \rrbracket \rightarrow \llbracket y, F^T \rrbracket)$

lemma $\llbracket x^T \preceq y^T \equiv (\exists z. (\langle x^T \oplus z^T \rangle = y^T)) \rrbracket = \top$ **apply simp oops**

lemma $\llbracket x^T \preceq y^T \equiv (x^T \oplus y^T = y^T) \rrbracket = \top$ **apply simp oops**

lemma $\llbracket \langle \lambda x. \langle R^T, x^T \rangle, y^T \rangle \rrbracket = X$ **apply simp oops**
lemma $\llbracket \langle y^T, \lambda x. \langle x^T, R^T \rangle \rrbracket = X$ **apply simp oops**
lemma $\llbracket \langle y^T, \lambda x. \langle R^T, x^T \rangle \rrbracket = X$ **apply simp oops**

References

- [1] C. Benzmüller and L. Paulson. Quantified multimodal logics in simple type theory. *Logica Universalis (Special Issue on Multimodal Logics)*, 7(1):7–20, 2013.
- [2] 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.
- [3] C. Benzmüller and B. Woltzenlogel Paleo. The inconsistency in Gödel's ontological argument: A success story for AI in metaphysics. In *IJCAI 2016*, 2016. Accepted for publication; to appear in 2016.
- [4] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.
- [5] E. N. Zalta. Principia metaphysica, a compilation of the theorems of the theory of abstract objects. Available at <https://mally.stanford.edu/publications.html>.
- [6] E. N. Zalta. *Abstract Objects: An Introduction to Axiomatic Metaphysics*. Dordrecht: D. Reidel, 1983.

- [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.