Positive Free Higher-Order Logic and its Automation via a Semantical Embedding

Irina Makarenko and Christoph Benzmüller

July 22, 2020

Positive Free Higher-Order Logic and its Automation via a Semantical Embedding

Irina Makarenko and Christoph Benzmüller

Freie Universität Berlin, Berlin, Germany
rna.mkr|c.benzmueller@gmail.com

Abstract. Free logics are a family of logics that are free of any existential assumptions. Unlike traditional classical and non-classical logics, they support an elegant modeling of nonexistent objects and partial functions as relevant for a wide range of applications in computer science, philosophy, mathematics, and natural language semantics. While free first-order logic has been addressed in the literature, free higher-order logic has not been studied thoroughly so far. The contribution of this paper includes (i) the development of a notion and definition of free higher-order logic in terms of a positive semantics (partly inspired by Farmer's partial functions version of Church's simple type theory), (ii) the provision of a faithful shallow semantical embedding of positive free higher-order logic into classical higher-order logic, (iii) the implementation of this embedding in the Isabelle/HOL proof-assistant, and (iv) the exemplary application of our novel reasoning framework for an automated assessment of Prior's paradox in positive free quantified propositional logics, i.e., a fragment of positive free higher-order logic.

Keywords: Knowledge representation and reasoning · Interactive and automated theorem proving · Philosophical foundations of AI · Partiality and undefinedness · Prior's paradox.

1 Introduction

The proper handling of nonexistence and partiality constitutes a key challenge not only for applications of formal methods in philosophy and mathematics but also for computational approaches to artificial intelligence and natural language [15, 17, 16]. In a so-called *free logic*, terms do not necessarily have to denote existing objects allowing for theories involving both partial and total functions. For that reason, free higher-order logics provide elegant solutions to the handling of some well-known paradoxes in knowledge representation and reasoning, many of which are beyond first-order logic. Moreover, free logics are well suited to represent abstract objects and to support hypothetical reasoning with fictive (and concrete) entities, and can therefore also be applied in metaphysics, ethics, and law.

Modern interactive and automated theorem provers, however, are typically developed for classical notions of logic, in which only total functions are supported natively. Instead of investing time and effort in the development of new theorem provers for free first-order and higher-order logics, a promising approach for the implementation of such logics in existing higher-order theorem provers are *shallow semantical embeddings* (SSEs) [5]. The contribution of this paper is fourfold: We (i) developed a notion and definition of free higher-order logic in terms of a positive semantics (partly inspired by Farmer's partial functions version of Church's simple type theory [14]), (ii) provided a faithful shallow semantical embedding of positive free higher-order logic into classical higher-order logic, (iii) implemented this embedding in the Isabelle/HOL proof-assistant, and (iv) applied our novel reasoning framework for an automated assessment of Prior's paradox [29] in positive free quantified propositional logics, i.e., a fragment of positive free higher-order logic. Furthermore, we are currently integrating the results reported in this paper in the LogiKEy framework [9] for expressive, pluralistic normative reasoning.

Prior, coinciding with Kaplan [19], showed that paradoxes can arise quickly in particular philosophical theories that include both sets and propositions. Bacon, Hawthorne, and Uzquiano [3] discovered that universal instantiation, or, better, the rejection of it, is key to blocking certain paradoxes inherent in such higher-order logics. Logics without existential assumptions, i.e., free logics, just naturally reject the principle of universal instantiation. The family of paradoxes considered by Bacon et al. is represented by what we will call Prior's paradox in this paper. Prior's paradox states:

$$Q\,\forall p.\ (Q\,p \rightarrow \neg p)\ \rightarrow\ \exists p.\ (Q\,p \wedge p) \wedge \exists p.\ (Q\,p \wedge \neg p)\,.$$

Reading $Q\,p$ as, e.g., 'Kaplan says at midnight that p', Prior's paradox implies that if Kaplan says at midnight that everything Kaplan says at midnight is false, then Kaplan has said a true and a false thing at midnight. We end up with a logical self-contradiction that, as we will discuss and demonstrate later in this paper, is indeed resolved in free higher-order logic.

The paper structure is as follows: Section 2 briefly recaps *classical higher-order logic* (HOL), before *positive free higher-order logic* (PFHOL) is introduced in Section 3. Section 4 presents a faithful embedding of PFHOL in HOL, and Section 5 discusses its encoding in Isabelle/HOL. Section 6 applies the encoded embedding to "solve" Prior's paradox, and the last section concludes the paper.

2 Classical Higher-Order Logic (HOL)

Church's *simple type theory* [13] is a classical higher-order logic defined on top of the simply typed λ-calculus. Church's original definitions, as generalized by Henkin [18] to *extensional type theory*, the logical basis of most automated theorem proving systems for higher-order logic, are summarized below.

2.1 Syntax

The main components of Church's type theory are types and terms; more precisely, typed terms. The set of *simple types* $\mathcal{T}$ is freely generated from a set of two base types, $\{o, i\}$, and the right-associative function type constructor $\rightarrow$. Intuitively, o is the type of standard truth values, and i is the type of individuals.[1] $\mathcal{T}$ is thus defined by $\alpha, \beta \coloneqq o \mid i \mid (\alpha \rightarrow \beta)$. $\mathcal{T}_o \subsetneq \mathcal{T}$, the set of *simple types of (goal) type* o, is given by $\beta \coloneqq o \mid (\alpha \rightarrow \beta)$ (with $\alpha \in \mathcal{T}$). $\mathcal{T}_i \subsetneq \mathcal{T}$, the set of *simple types of (goal) type* i, is analogously given by $\beta \coloneqq i \mid (\alpha \rightarrow \beta)$ (with $\alpha \in \mathcal{T}$).

Starting with some nonempty countable sets of typed constant symbols C_α and some nonempty countable sets of typed variable symbols V_α, the *simply typed terms* of HOL are defined by the following formation rules (where $\alpha, \beta \in \mathcal{T}$, $P_\alpha \in C_\alpha$, and $x_\alpha \in V_\alpha$):

$$s, t \coloneqq \quad P_\alpha \;\mid\; x_\alpha \;\mid\; (s_{\alpha\rightarrow\beta}\, t_\alpha)_\beta \;\mid\; (\lambda x_\alpha .\, s_\beta)_{\alpha\rightarrow\beta} \,.$$

We assume the following constant symbols to be part of our "signature": $\neg_{o\rightarrow o} \in C_{o\rightarrow o}$, $\vee_{o\rightarrow o\rightarrow o} \in C_{o\rightarrow o\rightarrow o}$, $=_{\alpha\rightarrow\alpha\rightarrow o} \in C_{\alpha\rightarrow\alpha\rightarrow o}$, $\forall_{(\alpha\rightarrow o)\rightarrow o} \in C_{(\alpha\rightarrow o)\rightarrow o}$, and $\iota_{(\alpha\rightarrow o)\rightarrow\alpha} \in C_{(\alpha\rightarrow o)\rightarrow\alpha}$ with $\alpha \in \mathcal{T}$. These constant symbols, which we call logical constants, have a fixed interpretation according to their intuitive meaning.[2] For example, the definite description $(\iota_{(\alpha\rightarrow o)\rightarrow\alpha}(\lambda x_\alpha .\, s_o)_{\alpha\rightarrow o})_\alpha$ denotes the unique object x of type $\alpha \in \mathcal{T}$ satisfying s_o if such an object exists and some fixed but arbitrary object of type α otherwise. It offers the possibility to define an if-then-else operator as follows (with $\alpha \in \mathcal{T}$):

$$ite_{o\rightarrow\alpha\rightarrow\alpha\rightarrow\alpha} \coloneqq \lambda s_o .\, \lambda x_\alpha .\, \lambda y_\alpha .\; \iota(\lambda z_\alpha .\, (s \rightarrow z = x) \wedge (\neg s \rightarrow z = y))\,.$$

Further logical constants can be introduced as abbreviations, e.g., $\wedge_{o\rightarrow o\rightarrow o} \coloneqq \lambda x_o .\, \lambda y_o .\, \neg(\neg x \vee \neg y)$ and $\exists_{(\alpha\rightarrow o)\rightarrow o} \coloneqq \lambda p_{\alpha\rightarrow o} .\, \neg\forall(\lambda x_\alpha .\, \neg(p\, x))$ with $\alpha \in \mathcal{T}$. Terms of type o are *formulas*, nonformula terms of type $\alpha \in \mathcal{T}_o$ are called *predicates*. Formulas whose leftmost nonparenthesis symbol is either equality or some nonlogical constant or variable are called *atomic formulas*. A variable x is *bound* in a term s if it occurs in the scope of the binder λ in s. x is *free* in s when it is not bound in s.

Type information may be omitted if clear from the context. For each binary operator *op* with prefix notation $((op\, s)\, t)$ we may fall back to its infix notation $(s\, op\, t)$ to improve readability. Likewise, the binder notation $\{\forall, \iota\}(x.\, s)$ may be used as shorthand for $\{\forall, \iota\}(\lambda x.\, s)$. In the remainder of this paper, a matching pair of parentheses in a type or term may be dropped when they are not necessary, assuming that, in addition to the generally known rules, $s\, t$, function application, and $\lambda x.\, s$, function abstraction, are left- and right-associative[3], respectively, and that application has a smaller scope than abstraction.

[1] There is no serious restriction to a two-valued base set so that further base types could be added [8].

[2] The set of primitive logical constants could be a much smaller one, e.g., equality is known to be sufficient in order to define all remaining logical constants of classical higher-order logic apart from the description operator [6].

[3] For an abstraction, being right-associative means that its body extends as far right as possible. For instance, $\lambda x. s\, t$ corresponds to $\lambda x.\, (s\, t)$ and not $(\lambda x.\, s)\, t$.

2.2 Semantics

A *frame* $D = \{D_\alpha : \alpha \in \mathcal{T}\}$ is a set of nonempty sets (or *domains*) D_α, such that D_i is chosen freely, $D_o = \{\mathrm{T}, \mathrm{F}\}$ where $\mathrm{T} \neq \mathrm{F}$ and T represents truth and F represents falsehood, and $D_{\alpha\to\beta}$ is the set of all total functions from domain D_α to codomain D_β. A *standard model* is a tuple $M = \langle D, I \rangle$ where D is a frame and I is a family of typed interpretation functions, i.e., $I = \{I_\alpha : \alpha \in \mathcal{T}\}$. Each *interpretation function* I_α maps constants of type α to appropriate objects of D_α. The logical constants $=$, $\neg$, $\vee$, $\forall$, and ι are interpreted as follows:

$$
\begin{array}{llll}
I(=_{\alpha\to\alpha\to o}) & := \mathit{id} & \in D_{\alpha\to\alpha\to o} & \text{s.t. for all } d, d' \in D_\alpha\text{:} \\
 & & & \quad \mathit{id}(d, d') = \mathrm{T} \text{ iff } d \text{ is identical to } d', \\
I(\neg_{o\to o}) & := \mathit{not} & \in D_{o\to o} & \text{s.t. } \mathit{not}(\mathrm{T}) = \mathrm{F} \text{ and } \mathit{not}(\mathrm{F}) = \mathrm{T}, \\
I(\vee_{o\to o\to o}) & := \mathit{or} & \in D_{o\to o\to o} & \text{s.t. } \mathit{or}(v_1, v_2) = \mathrm{T} \text{ iff } v_1 = \mathrm{T} \text{ or } v_2 = \mathrm{T}, \\
I(\forall_{(\alpha\to o)\to o}) & := \mathit{all} & \in D_{(\alpha\to o)\to o} & \text{s.t. for all } f \in D_{\alpha\to o}\text{:} \\
 & & & \quad \mathit{all}(f) = \mathrm{T} \text{ iff } f(d) = \mathrm{T} \text{ for all } d \in D_\alpha, \\
I(\iota_{(\alpha\to o)\to\alpha}) & := \mathit{desc} & \in D_{(\alpha\to o)\to\alpha} & \text{s.t. for all } f \in D_{\alpha\to o}\text{:} \\
 & & & \quad \mathit{desc}(f) = d \in D_\alpha \text{ if } f(d) = \mathrm{T} \text{ and for} \\
 & & & \quad \text{all } d' \in D_\alpha\text{: if } f(d') = \mathrm{T}, \text{ then } d' = d, \\
 & & & \quad \text{otherwise } \mathit{desc}(f) = e \text{ where } e \text{ is a} \\
 & & & \quad \text{fixed but arbitrary object in } D_\alpha.
\end{array}
$$

g_α is a *variable assignment* mapping variables of type α to corresponding objects in D_α. Thus, $g = \{g_\alpha : \alpha \in \mathcal{T}\}$ is a family of typed variable assignments. $g[x \to d]$ denotes the variable assignment that is identical to g, except for variable x_α, which is now mapped to d_α. The *value* $[\![s_\alpha]\!]^{M,g}$ of a HOL term s_α in a standard model M under variable assignment g is an object $d \in D_\alpha$ and defined as follows:

$$
\begin{array}{lll}
[\![P_\alpha]\!]^{M,g} & := & I(P_\alpha), \\
[\![x_\alpha]\!]^{M,g} & := & g(x_\alpha), \\
[\![(s_{\alpha\to\beta}\, t_\alpha)_\beta]\!]^{M,g} & := & [\![s_{\alpha\to\beta}]\!]^{M,g}([\![t_\alpha]\!]^{M,g}), \\
[\![(\lambda x_\alpha.\ s_\beta)_{\alpha\to\beta}]\!]^{M,g} & := & \text{the function } f \text{ from } D_\alpha \text{ into } D_\beta \\
 & & \text{s.t. for all } d \in D_\alpha\text{: } f(d) = [\![s_\beta]\!]^{M,g[x\to d]}.
\end{array}
$$

A formula s_o is *true* in a standard model M under variable assignment g, denoted by $M, g \vDash s$, if and only if $[\![s_o]\!]^{M,g} = \mathrm{T}$. A formula s_o is *valid in* M, denoted by $M \vDash s$, if and only if $M, g \vDash s$ for all variable assignments g. Moreover, a formula s_o is *(generally) valid*, denoted by $\vDash s_o$, if and only if s_o is valid in all standard models M.

As a consequence of Gödel's incompleteness theorem, Church's type theory with respect to the ordinary semantics based on standard models is incomplete. However, Henkin [18] introduced a generalized notion of a model in which the function domains contain enough but not necessarily all functions: In a standard model, a domain $D_{\alpha\to\beta}$ is defined as the set of all total functions from D_α to D_β. In a *Henkin model* (or *general model*) the domains $D_{\alpha\to\beta}$ in the underlying frame are some nonempty sets of total functions, $D_{\alpha\to\beta} \subseteq \{\, f \mid f : D_\alpha \to D_\beta \}$, containing at least sufficiently many of them such that the valuation function remains total.

For Henkin's generalized notion of semantics, sound and complete proof calculi exist [18, 1, 2]. Any standard model is obviously also a Henkin model. Hence, any formula that is valid in all Henkin models must be valid in all standard models as well. Therefore, the semantics employed in this paper are Henkin's general models. For truth, validity, and general validity in a Henkin model, the above definitions are adapted in the obvious way.

For further details on the semantics of HOL, we refer to the literature [7, 6].

3 Positive Free Higher-Order Logic (PFHOL)

Free logic, a term coined by Lambert [21], refers to a family of logics that are free of existential presuppositions in general and with respect to the denotation of terms in particular. Terms of free logic may denote existent[4] objects, but are not necessarily required to do so. Quantification and definite descriptions are treated as in classical logic, meaning that quantifiers and description operators range over the existing objects only. In the following, we will pursue an *inner-outer dual-domain approach* for the representation of the relationship between existing and nonexisting objects. The inner-outer dual-domain approach postulates that some domain D contains both existing and nonexisting objects whereas the quantification domain E, a subdomain of D, contains solely the existing ones.

A free logic is known to be positive if it allows atomic formulas containing terms that refer to nonexisting objects to be either true or false [32, 22]. For example, even though $isHuman(Pegasus)$ is, in general, denied, $hasLegs(Pegasus)$ may be regarded as a valid formula since the denotation of $Pegasus$ is a mythological creature that is usually depicted in the form of a winged horse (with legs).

3.1 Syntax

Except for terms, all definitions and terminology for PFHOL correspond to those presented in Section 2.1 for HOL. Simply typed terms of PFHOL have essentially the same structure as terms of HOL, but we additionally include the nonlogical constant symbol $E!_{\alpha\to o} \in C_{\alpha\to o}$ in the "signature". Apart from that, the interpretation of the universal quantifier changes since free logical quantification is traditionally limited to existing objects only. Moreover, not only quantifiers have existential import: Definite descriptions of free logic denote a unique object satisfying some property if and only if such an object exists and is defined [4].

[4] In the paper at hand, the terms existent/existing and defined are used interchangeably even though a differentiation is advisable. The same applies to the terms nonexistent/nonexisting and undefined.

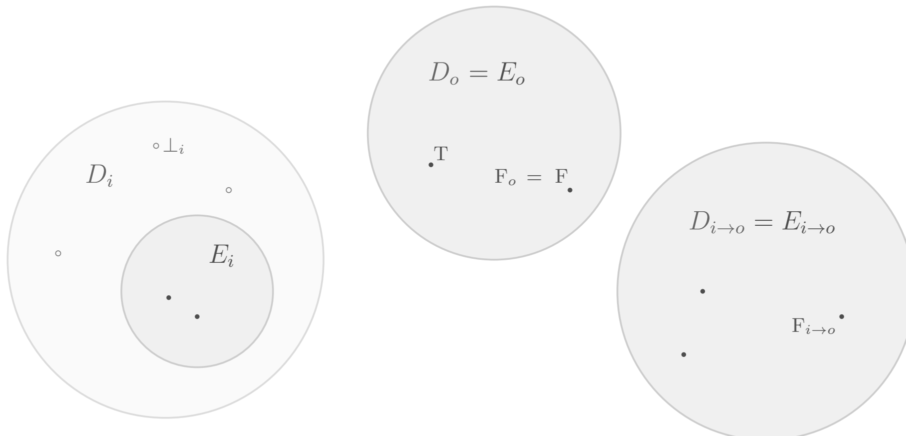

Fig. 1. Schematics of domains D_i, D_o and $D_{i\to o}$

3.2 Semantics

The following proposal of a positive semantics for free higher-order logic combines two sophisticated concepts that go back to Benzmüller and Scott [10] and Farmer [14].

While a frame is defined exactly as in HOL, a *subframe* $E = \{E_\alpha : \alpha \in \mathcal{T}\}$ is a set of nonempty sets (or *domains*) E_α such that $E_\alpha \subsetneq D_\alpha$ for each $\alpha \in \mathcal{T}_i$ and $E_\alpha = D_\alpha$ for each $\alpha \in \mathcal{T}_o$.[5] We assume, inspired by Farmer, that $\bot_\alpha \in D_\alpha \setminus E_\alpha$ for all $\alpha \in \mathcal{T}_i$ with $\bot_{\alpha\to\beta}(d) := \bot_\beta$ for all $d \in D_\alpha$. Furthermore, each domain D_α with $\alpha \in \mathcal{T}_o$ contains the element F_α defined inductively by $\mathrm{F}_o := \mathrm{F}$ and $\mathrm{F}_{\alpha\to\beta}(d) := \mathrm{F}_\beta$ for all $d \in D_\alpha$. The purpose of these objects is to propagate the nondenotation or falsehood of a term up through all terms containing it with $\bot_i$ symbolizing 'the undefinedness' among individuals. Their intended use will be explained in the further course of this section. Exemplary schematics of some of the domains can be found in Fig. 1. A *standard model* is a triple $M = \langle D, E, I \rangle$ where D is a frame, E is a subframe, and I is a family of typed interpretation functions, i.e., $I = \{I_\alpha : \alpha \in \mathcal{T}\}$. Each *interpretation function* I_α maps constants of type α to appropriate elements of D_α. The nonlogical constant *E!* and the logical constants $=$, $\neg$, $\vee$, $\forall$ and ι are interpreted as follows:

$$I(E!_{\alpha\to o}) := ex \in E_{\alpha\to o} \quad \text{s.t. for all } d \in D_\alpha\text{: } ex(d) = \mathrm{T} \text{ iff } d \in E_\alpha,$$

$$I(=_{\alpha\to\alpha\to o}) := id \in E_{\alpha\to\alpha\to o} \quad \text{s.t. for all } d, d' \in D_\alpha\text{: } id(d,d') = \mathrm{T} \text{ iff } d \text{ is identical to } d',$$

$$I(\neg_{o\to o}) := not \in E_{o\to o} \quad \text{s.t. } not(\mathrm{T}) = \mathrm{F} \text{ and } not(\mathrm{F}) = \mathrm{T},$$

[5] Restricting nondenotation to the domain of individuals, i.e., to define $E_i \subsetneq D_i$ and for all $\alpha \neq i$, $E_\alpha = D_\alpha$, is reasonable but complicates the definition of strict functions.

$$
\begin{array}{llll}
I(\vee_{o\to o\to o}) & := or & \in E_{o\to o\to o} & \text{s.t. } or(v_1, v_2) = \mathrm{T} \text{ iff } v_1 = \mathrm{T} \text{ or } v_2 = \mathrm{T}, \\
I(\forall_{(\alpha\to o)\to o}) & := all & \in E_{(\alpha\to o)\to o} & \text{s.t. for all } f \in D_{\alpha\to o}\text{:} \\
 & & & \quad all(f) = \mathrm{T} \text{ iff } f(d) = \mathrm{T} \text{ for all } d \in E_\alpha, \\
I(\iota_{(\alpha\to o)\to\alpha}) & := desc & \in E_{(\alpha\to o)\to\alpha} & \text{s.t. for all } f \in D_{\alpha\to o}\text{:} \\
 & & & \quad desc(f) = d \in E_\alpha \text{ if } f(d) = \mathrm{T} \text{ and for} \\
 & & & \quad \text{all } d' \in E_\alpha\text{: if } f(d') = \mathrm{T}\text{, then } d' = d, \\
 & & & \quad \text{otherwise } desc(f) = \bot_\alpha \text{ if } \alpha \in \mathcal{T}_i \text{ and} \\
 & & & \quad desc(f) = \mathrm{F}_\alpha \text{ if } \alpha \in \mathcal{T}_o.
\end{array}
$$

As for HOL, g_α is a *variable assignment* mapping variables of type α to corresponding objects in D_α. The value $[\![s_\alpha]\!]^{M,g}$ of a PFHOL term s_α in a standard model M under the variable assignment g is an object $d \in D_\alpha$ and evaluated as follows:

$$
\begin{array}{lll}
[\![P_\alpha]\!]^{M,g} & := & I(P_\alpha), \\
[\![x_\alpha]\!]^{M,g} & := & g(x_\alpha), \\
[\![(s_{\alpha\to\beta}\, t_\alpha)_\beta]\!]^{M,g} & := & [\![s_{\alpha\to\beta}]\!]^{M,g}([\![t_\alpha]\!]^{M,g}), \\
[\![(\lambda x_\alpha.\ s_\beta)_{\alpha\to\beta}]\!]^{M,g} & := & \text{the function } f \text{ from } D_\alpha \text{ into } D_\beta \\
 & & \text{s.t. for all } d \in D_\alpha\text{: } f(d) = [\![s_\beta]\!]^{M,g[x\to d]}.
\end{array}
$$

The application is hereby defined in a nonstrict manner. A strict function application would be defined like this (with $\alpha \to \beta \in \mathcal{T}_i$):

$$
[\![(s_{\alpha\to\beta}\, t_\alpha)_\beta]\!]^{M,g} \quad := \quad \begin{cases} [\![s_{\alpha\to\beta}]\!]^{M,g}([\![t_\alpha]\!]^{M,g}) & \text{if } [\![t_\alpha]\!]^{M,g} \in E_\alpha{}^{6} \\ \bot_\beta & \text{else}\,. \end{cases}
$$

A strictly applied function results in undefined if one of its arguments is undefined. In simple type theory, arguments are typically processed one after another. To be able to pass the undefined state of a once applied argument through any other possibly following arguments, the objects $\bot_\alpha$ were added to each relevant domain D_α. $\bot_{\alpha\to\beta}$ maps any argument of type α to $\bot_\beta$ until $\bot_i$ appears. This way, undefinedness is transmitted until the evaluation of the application has reached its end.[7] Predicates, on the other hand, do not generally require such a special treatment. In positive free logic, atomic formulas may denote truth or falsehood even if one of the arguments is undefined. Otherwise, the objects F_α could be used for transmitting falsehood.

The definitions of truth, validity, and general validity in PFHOL are equivalent to the corresponding definitions in HOL. The partiality characteristic for

[6] Farmer also checked the function itself for existence. But since the distinction between existing and nonexisting functions – in contrast to existing/nonexisting individuals – is unusual and not well-defined, this was left out.

[7] Restraining applications like this could lead to malformed evaluations, i.e., evaluated terms might not receive the actually intended value. For instance, the *ite* operator must be handled separately when the then- or else-parts are meant to be undefined.

free logic is implemented by a trick that exploits the objects $\bot_\alpha$, enabling the functions in each domain $D_{\alpha\to\beta}$ to remain total. Hence, the generalization of standard models to Henkin models is equally applicable to PFHOL.[8]

4 Embedding of PFHOL in HOL

To provide a shallow semantical embedding of PFHOL in HOL, the "signature" of HOL has to be enriched with an additional nonlogical constant $E_{\alpha\to o} \in C_{\alpha\to o}$ denoting a unary predicate that enables an explicit distinction of existing and nonexisting objects in the domain D_α. In addition, we include the object e_α in each domain D_α with $\alpha \in \mathcal{T}$, which is meant to be the error object that is returned by the definite description $(\iota_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha$ if no such object exists. We redefine the interpretation of ι thus as follows:

$$I(\iota_{(\alpha\to o)\to\alpha}) := desc \in D_{(\alpha\to o)\to\alpha} \text{ s.t. for all } f \in D_{\alpha\to o}\text{: } desc(f) = d \in D_\alpha \text{ if } f(d) = \mathrm{T} \text{ and for all } d' \in D_\alpha\text{: if } f(d') = \mathrm{T}, \text{ then } d' = d, \text{ otherwise } desc(f) = e_\alpha.$$

Obviously, for all $\alpha \in \mathcal{T}_o$: $(\forall x_\alpha.\,(E_{\alpha\to o}\, x_\alpha)_o)_o = \mathrm{T}$, and $(E_{\alpha\to o}\, e_\alpha)_o = \mathrm{F}$ for each $\alpha \in \mathcal{T}_i$. Then, a HOL term $[\, s_\alpha \,]$ is assigned to each PFHOL term s_α according to the following translation function:[9]

$$
\begin{aligned}
[\, P_\alpha \,] &= P_\alpha,\\
[\, x_\alpha \,] &= x_\alpha,\\
[\, (E!_{\alpha\to o}\, s_\alpha)_o \,] &= (E_{\alpha\to o}\, [\, s_\alpha \,])_o,\\
[\, ((=^F_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o \,] &= ((=^H_{\alpha\to\alpha\to o}\, [\, s_\alpha \,])_{\alpha\to o}\, [\, t_\alpha \,])_o,\\
[\, (\neg^F_{o\to o}\, s_o)_o \,] &= (\neg^H_{o\to o}\, [\, s_o \,])_o,\\
[\, ((\wedge^F_{o\to o\to o}\, s_o)_{o\to o}\, t_o)_o \,] &= ((\wedge^H_{o\to o\to o}\, [\, s_o \,])_{o\to o}\, [\, t_o \,])_o,\\
[\, (\forall^F_{(\alpha\to o)\to o}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_o \,] &= (\forall^H_{(\alpha\to o)\to o}(\lambda x_\alpha.\, ((E\, x)_o \to^H_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o})_o,\\
[\, (\iota^F_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha \,] &= (\iota^H_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, ((E\, x)_o \wedge^H_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o})_\alpha,\\
[\, (s_{\alpha\to\beta}\, t_\alpha)_\beta \,] &= ([\, s_{\alpha\to\beta} \,]\, [\, t_\alpha \,])_\beta,\\
[\, (\lambda x_\alpha.\, s_\beta)_{\alpha\to\beta} \,] &= (\lambda x_\alpha.\, [\, s_\beta \,])_{\alpha\to\beta}.
\end{aligned}
$$

Note that operators of HOL and PFHOL are annotated with ^{H} and ^{F}, respectively.

The main trick of this translation is that the existential import of the universal quantifier and the description operator is secured by cleverly exploiting

[8] As shown by Farmer and Schütte [31], it is possible to give a Henkin-style completeness proof for free higher-order logic defined based on a partial valuation function.

[9] A similar translation, although for free first-order logic, was provided and proved to be sound and complete by Meyer and Lambert [26] and Benzmüller and Scott [10].

the additional predicate $E_{\alpha\to o}$ as a guard. When mapping definite descriptions, $[\,(\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\ s_o)_{\alpha\to o})_\alpha\,]$ could also be translated into

$$
\begin{aligned}
&(ite^{H}_{o\to\alpha\to\alpha\to\alpha}\\
&\quad(\exists^{H}_{(\alpha\to o)\to o}(\lambda x_\alpha.\\
&\qquad(((E_{\alpha\to o}\,x_\alpha)_o \wedge^{H}_{o\to o\to o} [\,s_o\,])_o\\
&\qquad\ \wedge^{H}_{o\to o\to o} (\forall^{H}_{(\alpha\to o)\to o}(\lambda y_\alpha.(((E_{\alpha\to o}\,y_\alpha)_o \to^{H}_{o\to o\to o} [\,s_o\,])_o\\
&\qquad\qquad\qquad\qquad\to^{H}_{o\to o\to o} (y_\alpha =^{H}_{\alpha\to\alpha\to o} x_\alpha)_o)_o)_{\alpha\to o})_o)_o)_{\alpha\to o})_o\\
&\quad(\iota^{H}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\ ((E_{\alpha\to o}\,x_\alpha)_o \wedge^{H}_{o\to o\to o} [\,s_o\,])_o)_{\alpha\to o})_\alpha\\
&\quad e_\alpha)_\alpha
\end{aligned}
$$

using the if-then-else operator *ite* to ensure that the classical definite description definitely returns the error object e_α in case of no such existing object. But due to our previously done redefinition of the classical description operator, this is not really necessary here. Furthermore, it is noteworthy that any term $\exists^{F}x.\,s$ is translated into $\neg^{H}\forall^{H}x.\ E\,x \to^{H} \neg^{H}s$, which is the same as $\exists^{H}x.\ E\,x \wedge^{H} s$.

Next, we establish the faithfulness of this embedding.

Theorem 1. $\models_{PFHOL} s_o$ *if and only if* $\models_{HOL} [\,s_o\,]$.

The proof of Thm. 1 is sketched in the appendix. For full details, see Makarenko [25].

5 Implementation in Isabelle/HOL

In this section, the encoding of the embedding from Section 4 in Isabelle/HOL [27] is presented. The general syntax and semantics of Isabelle/HOL can be found in the specified literature and is therefore omitted here. The encoding starts with a declaration of the base type `i` for individuals while the type o of HOL is associated with the predefined type `bool` in Isabelle/HOL.

```
typedecl i
```

Next, we introduce an existence predicate, `E`, for each of the base and compound types. In the signature, the single quote in `'a` indicates that this is a type variable, meaning that the definition given hereupon is polymorphic.

```
consts fExistence :: "'a ⇒ bool" ("E")
```

Then, for each type, we define another new constant, **`e`**, and, in accordance with the definitions in Section 4, we postulate **`e`** of type `i` to be nonexistent and **`e`** of type `bool` to be `False`. Furthermore, `True` and `False` are declared as existent.

```
consts fUndef :: "'a" ("e")
axiomatization where fUndefIAxiom: "¬E (e::i)"
axiomatization where fFalsehoodBAxiom: "(e::bool) = False"
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"
```

The embedding of the logical constants $=$, $\neg$, and $\vee$ is straightforward. PFHOL operators are presented in bold-face fonts to distinguish them from HOL operators.

```
definition fIdentity :: "'a ⇒ 'a ⇒ bool" (infixr "=" 56)
    where "φ = ψ ≡ φ = ψ"
definition fNot :: "bool ⇒ bool" ("¬_" [52]53)
    where "¬φ ≡ ¬φ"
definition fOr :: "bool ⇒ bool ⇒ bool" (infixr "∨" 51)
    where "φ ∨ ψ ≡ φ ∨ ψ"
```

Now, for the embedding of the existential import of the universal quantifier, we utilize the existence predicate `E` of the respective type exactly as discussed in Section 4. Isabelle/HOL supports the introduction of syntactic sugar for binding notations, which we adopt in the following definition in order to support the more intuitive notation $\forall x.\, P\, x$ instead of writing $\forall(\lambda x.\, P\, x)$ or $\forall P$.

```
definition fForall :: "('a ⇒ bool) ⇒ bool" ("∀")
    where "∀Φ ≡ ∀x. E x ⟶ Φ x"
definition fForallBinder:: "('a ⇒ bool) ⇒ bool" (binder "∀" [8]9)
    where "∀x. φ x ≡ ∀φ"
```

For encoding the PFHOL operator ι, we rely on Isabelle/HOL's own definite description operator `THE`. Unlike the embedding from Section 4, we must here specify the object that will be returned if there is no unique object that has the desired properties. We use Isabelle/HOL's if-then-else operator for this.

```
definition fThat :: "('a ⇒ bool) ⇒ 'a" ("I")
  where "IΦ ≡ if ∃x. E x ∧ Φ x ∧ (∀y. (E y ∧ Φ y) ⟶ (y = x))
             then THE x. E x ∧ Φ x
             else e"
definition fThatBinder:: "('a ⇒ bool) ⇒ 'a" (binder "I" [8]9)
  where "Ix. φ x ≡ Iφ"
```

We also introduced binder notation for **I**. Further PFHOL operators are embedded as abbreviations.

```
definition fAnd :: "bool ⇒ bool ⇒ bool" (infixr "∧" 52)
  where "φ ∧ ψ ≡ ¬(¬φ ∨ ¬ψ)"
definition fImp :: "bool ⇒ bool ⇒ bool" (infixr "→" 49)
  where "φ → ψ ≡ ¬φ ∨ ψ"
definition fEquiv :: "bool ⇒ bool ⇒ bool" (infixr "↔" 50)
  where "φ ↔ ψ ≡ φ → ψ ∧ ψ → φ"
definition fExists :: "('a ⇒ bool) ⇒ bool" ("∃")
  where "∃Φ ≡ ¬(∀(λy. ¬(Φ y)))"
definition fExistsBinder :: "('a ⇒ bool) ⇒ bool" (binder "∃" [8]9)
  where "∃x. φ x ≡ ∃φ"
```

For experiments and tests, and for the Isabelle/HOL sources, see Makarenko [25].[10]

[10] The Isabelle/HOL sources are also available at https://github.com/stilleben/Free-Higher-Order-Logic.

6 Automated Assessment of Prior's Paradox

In our practical studies, we benefit from the fact that Isabelle/HOL integrates powerful reasoning tools such as the model finder Nitpick [11] and the meta-prover Sledgehammer [28], which, in turn, invokes third-party resolution provers, SMT solvers, and higher-order provers as Satallax [12] and Leo-III [34]. Applying Sledgehammer together with our embedding of PFHOL in HOL to Prior's paradox, we end up with the following result.

```
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"

lemma "(Q (∀p. (Q p ⟶ (¬p)))) ⟶ ((∃p. Q p ∧ p) ∧ (∃p. Q p ∧ (¬p)))"
  using Defs by (smt fFalseAxiom fTrueAxiom)
```

The theorem is valid. But as can be clearly seen, the theorem is proved by using the axioms `fTrueAxiom` and `fFalseAxiom` imposing that both truth values are defined. We try it again without these.

```
lemma "(Q (∀p. (Q p ⟶ (¬p)))) ⟶ ((∃p. Q p ∧ p) ∧ (∃p. Q p ∧ (¬p)))"
  nitpick [user_axioms=true, show_all, format=2]
  oops
```

```
Nitpick found a counterexample for card i = 3:

  Free variable:
    Q = (λx. _)(True := True, False := True)
  Constants:
    E = (λx. _)(True := True, False := False)
    E = (λx. _)(i1 := False, i2 := False, i3 := True)
    e = i2
    e = False
```

This time the model finder Nitpick actually found a countermodel. Observe that in this countermodel one of the two truth values is undefined, namely `False`. This coincides with the countermodel provided by Bacon, Hawthorne, and Uzquiano. However, on a metaphysical level, it is highly questionable to shift even one of the truth values into the undefined range. Bacon et al. themselves did not find this approach for overcoming the paradox very promising and have constructed other countermodels as a substitute, which we could not reproduce with our embedding of PFHOL in HOL. For these countermodels, at least three different truth values are needed, and hence trivalent or other many-valued free higher-order logics should be used for that. Research has already been conducted in this direction, which, so far, has concentrated mainly on using deep embeddings [35] as opposed to adapting shallow ones [33].

An alternative option, already explored and implemented by Makarenko [25], is to embed and automate the free semantics specially developed by Bacon et al. to overcome this particular paradox. The semantical theory they introduced is a positive free higher-order logic based on set theory where only (possible) worlds are taken as primitive, and the validity of propositions is then modeled as world dependent. The embedding of this 'modal' positive free logic has proved useful and adequate in dealing with the paradox, as was confirmed by verifying further, more reasonable countermodels to Prior's paradox. Moreover, it is worth mentioning that there is currently a growing interest to further adapt the definitions of Section 3 and the embedding of Section 4 to develop proper notions of modal and intensional positive free higher-order logic and to embed them faithfully in HOL. An interesting application, and related ongoing work, includes the exploitation of free logic machinery in Kirchner's embedding of hyperintensional second-order modal logic and abstract object theory in Isabelle/HOL [20, Footnote 7 and Section 5] utilized for the encoding, assessment, and further investigation of Zalta's *Principia Logico-Metaphysica* [36].

7 Conclusion

Positive free higher-order logic and its characteristics of nonexistent objects and partial functions have been faithfully represented in an adequately modified version of simple type theory. A key point of the inner-outer dual-domain approach is that partiality is only simulated instead of inherently accomodating it, such that a classical logic environment could be maintained. Subsequently, our embedding was implemented in Isabelle/HOL to support interactive and automated reasoning. We applied this embedding to Prior's paradox and reconstructed some of the results Bacon, Hawthorne, and Uzquiano provided in dealing with the theorem. This shows that certain paradoxes can fruitfully be addressed in free higher-order logic. However, we were also able to verify that two-valued free logic is not enough to resolve the issue. Our ongoing research has therefore also been concerned with other variants of free logic. Traditionally, the family of free logics involves not only positive free logic, but also negative [30], neutral [24], and supervaluational [4] free logic whose semantics differ in the way how atomic formulas with terms that refer to nonexistent objects are treated. Furthermore, free many-valued logic or a logic with more than one notion and/or degree of nonexistence could be imagined. Some of these variants have already been successfully embedded and tested in Isabelle/HOL, as for example negative free higher-order logic and partly also supervaluational free higher-order logic [25], others are still under development. Of special interest are in particular neutral free higher-order logic and, as indicated in the previous section, many-valued (positive) free higher-order logic. Obviously, a mixture between shallow and deep embedding appears conceivable in this context and worth investigating. Fact is, nondenoting terms have always been and will always be an intriguing subject in logic, and, considering the lack of theorem provers for free logic, the development of an appropriate definition of free logic suited for embedding in HOL as well as the automation of free logic via a semantical embedding seems more important than ever.

Acknowledgments. We thank the anonymous reviewers whose insightful comments and suggestions have helped to improve this manuscript.

Appendix

For the proof of Theorem 1, we first need to elaborate how to transform a PFHOL model M into a HOL model M^*, and a PFHOL variable assignment g into a HOL variable assignment g^*. We assume that $D^*_\alpha = D_\alpha$ and $C^*_\alpha \setminus \{E_{\alpha\to o}\} = C_\alpha \setminus \{E!_{\alpha\to o}\}$ for all $\alpha \in \mathcal{T}$, and set $e_\alpha = \bot_\alpha$ for each $\alpha \in \mathcal{T}_i$ and $e_\alpha = \mathrm{F}_\alpha$ for each $\alpha \in \mathcal{T}_o$. Then, $M = \langle D, E, I \rangle$ corresponds to the model $M^* = \langle D^*, I^* \rangle$ where I^* is a family of interpretation functions that assigns the standard interpretation to the logical constants $=$, $\neg$, $\vee$, $\forall$ and ι of HOL as described in Section 2. For all other constants $P_\alpha \neq E_{\alpha\to o}$, $P_\alpha \in C^*_\alpha$: $I^*(P_\alpha) = I(P_\alpha)$. The nonlogical constant $E_{\alpha\to o} \in C^*_\alpha$ is interpreted as follows:

$$I^*(E_{\alpha\to o}) \quad := \; ex \quad \in D^*_{\alpha\to o} \qquad \text{s.t. for all } d \in D^*_\alpha : \; ex(d) = \mathrm{T} \text{ iff } d \in E_\alpha.$$

We further assume $V^*_\alpha = V_\alpha$ for all $\alpha \in \mathcal{T}$, and hence, for all $x_\alpha \in V^*_\alpha$ and $\alpha \in \mathcal{T}$, $g^*_\alpha(x_\alpha) = g_\alpha(x_\alpha)$.

Next, we first need to establish the following lemma.

Lemma 1. *For all PFHOL models M and PFHOL variable assignments g,*

$$[\![s_\alpha]\!]^{M,g} = [\![[s_\alpha]]\!]^{M^*, g^*}.$$

The detailed proof of this lemma can be found in Makarenko [25].

Theorem 1. $\models_{PFHOL} s_o$ *if and only if* $\models_{HOL} [s_o]$.

Proof.

($\to$) The proof is by contraposition:

Assume $\not\models_{PFHOL} s_o$. Then, there exists a PFHOL model M and a variable assignment g such that $[\![s_o]\!]^{M,g} = \mathrm{F}$. By Lemma 1, $[\![s_o]\!]^{M,g} = [\![[s_o]]\!]^{M^*, g^*} = \mathrm{F}$. Hence, $\not\models_{HOL} [s_o]$.

($\leftarrow$) Analogous to above by contraposition and Lemma 1.

Therefore, the embedding of PFHOL in HOL is sound and complete.

References

1. Peter B. Andrews. General Models and Extensionality. *Journal of Symbolic Logic*, 37(2):395–397, 1972.
2. Peter B. Andrews. General Models, Descriptions, and Choice in Type Theory. *Journal of Symbolic Logic*, 37(2):385–394, 1972.
3. Andrew Bacon, John Hawthorne, and Gabriel Uzquiano. Higher-Order Free Logic and the Prior-Kaplan Paradox. *Canadian Journal of Philosophy*, 46(4-5):493–541, 2016.
4. Ermanno Bencivenga. Free Logics. In Dov M. Gabbay and Franz Günthner, editors, *Handbook of Philosophical Logic. Volume III: Alternatives in Classical Logic*, pages 373–426. Springer Netherlands, Dordrecht, 1986.
5. Christoph Benzmüller. Universal (Meta-)Logical Reasoning: Recent Successes. *Science of Computer Programming*, 172:48–62, 2019.
6. Christoph Benzmüller and Peter B. Andrews. Church's Type Theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition, 2019.
7. Christoph Benzmüller, Chad E. Brown, and Michael Kohlhase. Higher-Order Semantics and Extensionality. *Journal of Symbolic Logic*, 69(4):1027–1088, 2004.
8. Christoph Benzmüller and Dale Miller. Automation of Higher-Order Logic. In Dov M. Gabbay, Jörg H. Siekmann, and John Woods, editors, *Computational Logic*, volume 9 of *Handbook of the History of Logic*, pages 215–254. Elsevier, North Holland, 2014.
9. Christoph Benzmüller, Xavier Parent, and Leendert van der Torre. Designing Normative Theories for Ethical and Legal Reasoning: LogiKEy Framework, Methodology, and Tool Support. *Artificial Intelligence*, 287:103348, 2020.
10. Christoph Benzmüller and Dana Scott. Automating Free Logic in HOL, with an Experimental Application in Category Theory. *Journal of Automated Reasoning*, pages 1–20, 2019.
11. Jasmin Christian Blanchette and Tobias Nipkow. Nitpick: A Counterexample Generator for Higher-Order Logic Based on a Relational Model Finder. In Matt Kaufmann and Lawrence C. Paulson, editors, *First International Conference on Interactive Theorem Proving*, volume 6172 of *Lecture Notes in Computer Science*, pages 131—146, Edinburgh, United Kingdom, Jul 11–14 2010. Springer Berlin, Heidelberg.
12. Chad E. Brown. Satallax: An Automatic Higher-Order Prover. In *Proceedings of the 6th International Joint Conference on Automated Reasoning*, IJCAR'12, pages 111–117, Berlin, Heidelberg, 2012. Springer.
13. Alonzo Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(2):56–68, 1940.
14. William M. Farmer. A Partial Functions Version of Church's Simple Theory of Types. *Journal of Symbolic Logic*, 55:1269–1291, 1990.
15. Solomon Feferman. Logics for Termination and Correctness of Functional Programs. In Yiannis N. Moschovakis, editor, *Logic from Computer Science*, pages 95–127, New York, NY, 1992. Springer.
16. Raymond D. Gumb. Free Logic in Program Specification and Verification. In Edgar Morscher and Alexander Hieke, editors, *New Essays in Free Logic. In Honour of Karel Lambert*, volume 23, page 157–93. Springer Netherlands, Dordrecht, 2001.
17. Raymond D. Gumb and Karel Lambert. Definitions in Nonstrict Positive Free Logic. *Modern Logic*, 7(1):25–55, 1997.

18. Leon Henkin. Completeness in the Theory of Types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.
19. David Kaplan. A Problem in Possible Worlds Semantics. In Walter Sinnott-Armstrong, Diana Raffman, and Nicholas Asher, editors, *Modality, Morality and Belief: Essays in Honor of Ruth Barcan Marcus*, pages 41–52. Cambridge University Press, 1995.
20. Daniel Kirchner, Christoph Benzmüller, and Edward N. Zalta. Mechanizing Principia Logico-Metaphysica in Functional Type Theory. *Review of Symbolic Logic*, 13(1):206–218, 2020.
21. Karel Lambert. The Definition of E! in Free Logic. *Abstracts: The International Congress for Logic, Methodology and Philosophy of Science*, 1960.
22. Karel Lambert. Free Logic and the Concept of Existence. *Notre Dame Journal of Formal Logic*, 1-2(8):133–144, 1967.
23. Karel Lambert. *Philosophical Applications of Free Logic*. Oxford University Press, 1991.
24. Scott Lehmann. 'No Input, No Output' Logic. In Edgar Morscher and Alexander Hieke, editors, *New Essays in Free Logic. In Honour of Karel Lambert*, volume 23, pages 147–155. Springer Netherlands, Dordrecht, 2001.
25. Irina Makarenko. Free Higher-Order Logic – Notion, Definition and Embedding in HOL. Master's thesis, Freie Universität Berlin, 2020.
26. Robert K. Meyer and Karel Lambert. Universally Free Logic and Standard Quantification Theory. *Journal of Symbolic Logic*, 33(1):8–26, 1968.
27. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. Isabelle/HOL — A Proof Assistant for Higher-Order Logic. `http://isabelle.in.tum.de/doc/tutorial.pdf`, 2019. Last accessed on December 30, 2019.
28. Lawrence C. Paulson and Jasmin Christian Blanchette. Three Years of Experience with Sledgehammer, a Practical Link between Automatic and Interactive Theorem Provers. *Proceedings of the 8th International Workshop on the Implementation of Logics*, pages 131—146, 2015.
29. Arthur N. Prior. On a Family of Paradoxes. *Notre Dame Journal of Formal Logic*, 2(1):16–32, 1961.
30. Rolf Schock. *Logics Without Existence Assumptions*. Almqvist & Wiksell, Stockholm, 1968.
31. Kurt Schütte. Syntactical and Semantical Properties of Simple Type Theory. *Journal of Symbolic Logic*, 25(4):305–326, 1960.
32. Dana Scott. Existence and Description in Formal Logic. In Ralph Schoenman, editor, *Bertrand Russell: Philosopher of the Century*, pages 181–200. Little, Brown & Company, Boston, 1967. Repr. in [23], pp. 28–48.
33. Alexander Steen and Christoph Benzmüller. Sweet SIXTEEN: Automation via Embedding into Classical Higher-Order Logic. *Logic and Logical Philosophy*, 25(4):535–554, 2016.
34. Alexander Steen and Christoph Benzmüller. The Higher-Order Prover Leo-III. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning. IJCAR 2018*, volume 10900 of *Lecture Notes in Computer Science*, pages 108–116. Springer, 2018.
35. Jørgen Villadsen and Anders Schlichtkrull. Formalization of Many-Valued Logics. In Henning Christiansen, M. Dolores Jiménez-López, Roussanka Loukanova, and Lawrence S. Moss, editors, *Partiality and Underspecification in Information, Languages, and Knowledge*, pages 219–256. Cambridge Scholars Press, 2017.
36. Edward N. Zalta. Principia Logico-Metaphysica. `http://mally.stanford.edu/principia.pdf`. Draft/Excerpt; last accessed on May 31, 2020.