Proof-Theoretic Conservative Extension of HOL with Ad-hoc Overloading

Arve Gengelbach(✉) and Tjark Weber(✉)

Department of Information Technology, Uppsala University, Uppsala, Sweden
{arve.gengelbach,tjark.weber}@it.uu.se

Abstract. Logical frameworks are often equipped with an extensional mechanism to define new symbols. The definitional mechanism is expected to be conservative, i.e. it shall not introduce new theorems of the original language. The theorem proving framework Isabelle implements a variant of higher-order logic where constants may be ad-hoc overloaded, allowing a constant to have different definitions for non-overlapping types. In this paper we prove soundness and completeness for the logic of Isabelle/HOL with general (Henkin-style) semantics, and we prove model-theoretic and proof-theoretic conservativity for theories of definitions.

Keywords: Classical higher-order logic · Conservative theory extension · Proof-theoretic conservativity · Ad-hoc overloading · Isabelle

1 Introduction

With the help of theorem provers such as HOL4 [14] and Isabelle [13], users formalise mathematical reasoning to create machine-checkable proofs. For convenience and abstraction, these systems allow the user to define new types and constants. Definitions extend the theory that is underlying a formalisation with additional axioms, but they are expected to be *proof-theoretically conservative* (and Wenzel [16] adds further requirements). Informally, any formula that is derivable from the extended theory but expressible prior to the definition should be derivable also from the original theory. As a special case, extension by definitions shall preserve *consistency*; any extension by a definition shall not enable the derivation of a contradiction. Design flaws in definitional mechanisms have repeatedly led to inconsistencies in theorem provers [12,16].

In this paper we establish proof-theoretic conservativity for higher-order logic (HOL) with ad-hoc overloading, meaning that constant symbols may have different definitions for non-overlapping types. Proof-theoretic conservativity is not obvious in this setting [12]: type and constant definitions may depend on one another, and type definitions cannot be unfolded in the same way as constant definitions. Moreover, type and constant symbols may be declared and used, e.g. to define other symbols, before a definition for them is given.

V. K. I. Pun et al. (Eds.): ICTAC 2020, LNCS 12545, pp. 23–42, 2020.
https://doi.org/10.1007/978-3-030-64276-1_2

In this setting the notion of proof-theoretic conservativity as informally described above does not hold. Consider a theory extension that defines a previously declared constant symbol: the equational axiom that is introduced by the definition is expressible, but in general not derivable prior to the definition. Therefore, we consider a notion of proof-theoretic conservativity that separates signature extensions (declarations) from theory extensions (definitions). We prove that any formula that is derivable from a definitional extension but *independent* of the newly defined symbol is also derivable prior to the extension.

Our results are particularly relevant to the foundations of Isabelle/HOL [13], which is an implementation of the logic and definitional mechanisms considered in this paper. As a practical consequence, proof-theoretic conservativity allows independence to be used as a syntactic criterion for deciding [8] whether a definition may be ignored when searching for the proof of a formula.

We establish proof-theoretic conservativity via several intermediate results that are of interest in their own right. Proof-theoretic conservativity has a semantic counterpart, *model-theoretic conservativity.* It is known [5,6] that HOL with ad-hoc overloading satisfies model-theoretic conservativity w.r.t. *lazy ground semantics*, that is a semantics that defines interpretations only for ground (i.e. type-variable free) symbols and for which semantic entailment is quantified over term variable valuations parametrised by ground type instantiation. In this paper, we generalise the model-theoretic conservativity result to a *general* (Henkin-style) [7] version of the lazy ground semantics. Subsequently, we prove the HOL deductive system sound and complete for general semantics. Soundness and completeness imply that provability and semantic validity coincide; thus we obtain proof-theoretic conservativity from model-theoretic conservativity.

Proof-theoretic conservativity of HOL with ad-hoc overloading has been studied before, notably by Kunčar and Popescu [9], who showed proof-theoretic conservativity of definitional theories relative to a fixed minimal theory, called *initial HOL*. Our result is stronger: we show proof-theoretic conservativity of definitional extensions relative to an arbitrary definitional theory. From this, we can immediately recover Kunčar and Popescu's result by noting that initial HOL is definitional; but our conservativity result also applies to formulae that do not belong to the language of initial HOL.

Contributions. We make the following contributions:

– We define a Henkin-style [7] generalisation of the lazy ground semantics [1] for HOL by relaxing the interpretation of function types to subsets of the set-theoretic function space (Sect. 4).
– We show model-theoretic conservativity for HOL with ad-hoc overloading w.r.t. general semantics (Sect. 5.1) by adapting an earlier proof of model-theoretic conservativity for lazy ground semantics [5,6].
– We prove soundness (Sect. 5.2) and completeness (Sect. 5.3) of the HOL deductive system for general semantics. The former is proved by induction over derivations, the latter by adapting ideas from Henkin's original completeness proof for the theory of types [7].

– From model-theoretic conservativity, soundness and completeness we derive a proof-theoretic conservativity result for HOL with ad-hoc overloading (Sect. 5.4). This result generalises the conservativity result of Kunčar and Popescu [9].

2 Related Work

We briefly review related work on meta-theoretical properties of higher-order logic for both standard and Henkin semantics, and discuss the different approaches to consistency and conservativity of HOL with ad-hoc overloading. Finally, we outline how our own work connects to these results.

Higher-Order Logic and Standard Semantics. HOL extends the *simple theory of types* introduced by Church [4] with rank-1 polymorphism. The logic is implemented, e.g. in HOL4 [14] and Isabelle/HOL [13], with slightly different definitional mechanisms. In Isabelle/HOL, ad-hoc overloading is a feature of the logic; in HOL4, it is supported through extensions of parsing and pretty-printing.

Pitts [15, § 15.4] introduces standard models for HOL (without ad-hoc overloading). These are set-theoretic models of theories whose signature contains the constants $\Rightarrow_{\mathsf{bool}\to\mathsf{bool}\to\mathsf{bool}}$, $\dot{=}_{\alpha\to\alpha\to\mathsf{bool}}$ and $\mathsf{some}_{(\alpha\to\mathsf{bool})\to\alpha}$ that are interpreted as logical implication, equality and Hilbert-choice, respectively. Types are inhabited, i.e. their interpretation is a non-empty set, and function types are interpreted by the corresponding set-theoretic function space.

This semantics and its properties in interplay with the deductive system are discussed in detail in the HOL4 documentation. For standard semantics HOL is sound [14, § 2.3.2] but incomplete, as by Gödel's incompleteness theorem there are unprovable sequents that hold in all standard models.

Henkin Semantics. Despite its incompleteness for standard semantics, HOL is complete for *Henkin semantics* [7], which relaxes the notion of a model and allows function types to be interpreted by proper subsets of the corresponding function space. For this semantics Andrews [2] proves completeness of a monomorphic logic $\mathcal{Q}_0$ whose built-in symbols are expressed in terms of equality. Andrews approaches the completeness proof as follows.

Each (syntactically) consistent theory is contained in a *maximal* consistent set of formulae, i.e. adding any other formula to the set would render it inconsistent. For maximal consistent theories one can construct a model by defining a term's interpretation as the equivalence class of provably equal terms. Hence, every consistent theory has a model (as there is a model of a maximal consistent superset of that theory). Completeness follows by contraposition.

Both Andrews and Henkin discuss completeness of variants of the simple theory of types. Their results transfer to HOL with Henkin-style semantics.

HOL with Ad-hoc Overloading. Wenzel [16] considers proof-theoretic conservativity “a minimum requirement for wellbehaved extension mechanisms.” He introduces the stronger notion of *meta-safety*, which additionally requires that new constants are syntactically realisable, i.e. they can be replaced with a term in the language of the original theory while preserving provability.

Kunčar and Popescu [9] extend meta-safety to type definitions and show meta-safety of definitional theories over *initial HOL*, the theory of Booleans with Hilbert-choice and the axiom of infinity. Their result is achieved by unfolding constant definitions and removing type definitions via a relativisation predicate over the corresponding host type. It follows that any definitional theory is a proof-theoretically conservative extension of initial HOL, and (as a special case) that any definitional theory is consistent. This result regards definitional theories as monolithic extensions of initial HOL, despite the incremental nature of theory extension by iterated application of definitional mechanisms.

The same authors prove definitional theories consistent by a semantic argument [12], introducing a so-called *ground* semantics for HOL, where validity of formulae is quantified over ground type instantiations. Hence ground semantics only ever interprets type-variable free terms. Given a definitional theory, Kunčar and Popescu construct a model for ground semantics by recursion over a well-founded *dependency relation* that relates each definiendum (i.e. each left-hand side of a definition) to any symbol occurring in the definiens. Åman Pohjola and Gengelbach [1] mechanise this model construction in HOL4 and correct fixable mistakes, leading to a larger dependency relation and a term semantics that applies type substitutions *lazily*, i.e. at the latest possible moment when interpreting a term.

In earlier work [6], we show model-theoretic conservativity of HOL with ad-hoc overloading by refining Kunčar and Popescu’s monolithic model construction (which builds a model from the ground up) to instead extend a given model of a definitional theory. Gengelbach et al. [5], again using lazy ground semantics, mechanise this result in HOL4.

Connection to Our Work. This paper combines some of the mentioned related work in new ways. First, we generalise lazy ground semantics in the sense of Henkin, relaxing the interpretation of function types. The deductive system of HOL is sound and complete with respect to general (Henkin-style) lazy ground semantics. For the proof of completeness, we adapt ideas from Henkin’s and Andrews’s completeness proofs to account for polymorphism. Second, re-using and extending our earlier proof of model-theoretic conservativity, we strengthen this result to general (Henkin-style) lazy ground semantics. Third, we combine soundness, completeness, and model-theoretic conservativity to obtain proof-theoretic conservativity of HOL with ad-hoc overloading relative to an arbitrary definitional theory, thereby generalising the conservativity result of Kunčar and Popescu.

3 Background

We introduce the language of polymorphic higher-order logic (HOL) (Sect. 3.1), definitional theories (Sect. 3.2), and the deductive system (Sect. 3.3). Parts of this section are adapted from our previous work [6, §2]. Our notation and terminology largely agree with other [1,5,12] related work.

3.1 The Language of Polymorphic HOL

The syntax of polymorphic HOL is that of the simply-typed lambda calculus, enriched with a first-order language of types. We fix two infinite sets TVar of *type variables*, ranged over by α, β, and Var of *term variables*, ranged over by x, y.

Signatures. A *signature* is a quadruple $(K, \mathrm{arOf}, \mathsf{Const}, \mathrm{tpOf})$, where K and Const are two countably infinite, disjoint sets. The elements of K are *type constructors*, and those of Const are *constants*. Each type constructor has an associated *arity*, given by the function $\mathrm{arOf} : K \to \mathbb{N}$. Each constant has an associated *type*, given by the function $\mathrm{tpOf} : \mathsf{Const} \to \mathsf{Type}$, where the set Type, ranged over by σ, τ, is defined inductively as the smallest set such that

- $\mathsf{TVar} \subseteq \mathsf{Type}$, and
- $(\sigma_1, \ldots, \sigma_n)k \in \mathsf{Type}$ whenever $k \in K$, $\mathrm{arOf}(k) = n$ and $\sigma_1, \ldots, \sigma_n \in \mathsf{Type}$.

For technical reasons (cf. Lemma 2), we assume that Const contains infinitely many constants a with $\mathrm{tpOf}(\mathsf{a}) \in \mathsf{TVar}$.

Built-in Types and Constants. For the remainder of this paper, we will assume a fixed signature. Moreover, we assume that K contains the following *built-in* type constructors: bool of arity 0, ind of arity 0, and $\to$ a right-associative type constructor of arity 2. A type is *built-in* if its type constructor is.

We also assume that Const contains the following *built-in* constants:

- $\Rightarrow$ of type $\mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$,
- $\doteq$ of type $\alpha \to \alpha \to \mathsf{bool}$,
- some of type $(\alpha \to \mathsf{bool}) \to \alpha$,
- zero of type ind,
- succ of type $\mathsf{ind} \to \mathsf{ind}$.

Instances. A *type substitution* is a function $\rho\colon \mathsf{TVar} \to \mathsf{Type}$ that replaces type variables by types. We extend type substitutions homomorphically to all types, and denote the set of type substitutions by TSubst. For any $\rho \in \mathsf{TSubst}$ and any type $\sigma \in \mathsf{Type}$, $\rho(\sigma)$ is a *(type) instance* of σ, written $\rho(\sigma) \leq \sigma$.

The set of *constant instances* CInst is a subset of the cartesian product $\mathsf{Const} \times \mathsf{Type}$ that contains exactly those tuples $(c, \sigma) \in \mathsf{CInst}$ for which the type σ is a type instance of the type of c, namely $\sigma \leq \mathrm{tpOf}(c)$. We use c_σ as shorthand notation for the tuple (c, σ). A constant instance is *built-in* if it is an instance of a built-in constant. For example, the constant instances $\doteq_{\mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}}$ and $\doteq_{\mathsf{ind} \to \mathsf{ind} \to \mathsf{bool}}$ are built-in as both are instances of the built-in constant $\doteq$, which has the type $\mathrm{tpOf}(\doteq) = \alpha \to \alpha \to \mathsf{bool}$.

Terms. The *terms* of our language are given by the following grammar, where x ranges over term variables, σ and τ are types, and c_σ ranges over constant instances:

$$t, t' \quad ::= \quad x_\sigma \quad | \quad c_\sigma \quad | \quad (t_{\sigma \to \tau}\ t'_\sigma)_\tau \quad | \quad (\lambda x_\sigma.\, t_\tau)_{\sigma \to \tau}$$

We may write t for t_σ when there is no risk of ambiguity. We require all terms to be *well-typed*, e.g. in $t\ t'$ the type of t' equals the argument type of t. Equality of terms is considered modulo α-equivalence. The set of all terms is denoted by Term.

A term is *closed* if it does not contain any free (term) variables. We extend tpOf to terms by defining $\text{tpOf}(t_\sigma) := \sigma$. Terms of type bool are called *formulae.*

For $u \in \mathsf{Term} \cup \mathsf{Type}$, we write $\text{TV}(u)$ for the set of type variables that occur syntactically in u. We can apply a type substitution ρ to a term t, written $\rho(t)$, by applying ρ to all type variables that occur in t.

Non-Built-Ins. To obtain the immediate non-built-in sub-types of a type, we define the following function $\cdot^\bullet$ on types:

$$\alpha^\bullet := \{\alpha\} \qquad \mathsf{bool}^\bullet := \emptyset \qquad \mathsf{ind}^\bullet := \emptyset \qquad (\sigma \to \tau)^\bullet := \sigma^\bullet \cup \tau^\bullet$$
$$((\sigma_1, \ldots, \sigma_n)\, k)^\bullet := \{(\sigma_1, \ldots, \sigma_n)\, k\} \text{ for } k \notin \{\mathsf{bool}, \to, \mathsf{ind}\}$$

For instance, if K contains a unary type constructor list, $(\alpha\ \mathsf{list} \to \mathsf{bool})^\bullet = \{\alpha\ \mathsf{list}\}$. We overload $\cdot^\bullet$ for terms, to collect non-built-in types of terms:

$$x_\sigma{}^\bullet := \sigma^\bullet \qquad c_\sigma{}^\bullet := \sigma^\bullet \qquad (s\ t)^\bullet := s^\bullet \cup t^\bullet \qquad (\lambda x.\, t)^\bullet := x^\bullet \cup t^\bullet$$

The *non-built-in types* of $M \subseteq \mathsf{Type}$ are those types for which $\cdot^\bullet$ is invariant, i.e. $M^\bullet := \{x \in M \mid x^\bullet = \{x\}\}$.

The operator $\cdot^\circ$ collects all non-built-in constant instances in a term:

$$x_\sigma{}^\circ := \emptyset \quad c_\sigma{}^\circ := \begin{cases} \emptyset & \text{if } c \text{ is built-in} \\ \{c_\sigma\} & \text{otherwise} \end{cases} \quad (s\ t)^\circ := s^\circ \cup t^\circ \quad (\lambda x.\, t)^\circ := t^\circ$$

Similar as defined for types, the *non-built-in constant instances* of $M \subseteq \mathsf{CInst}$ are those constant instances for which $\cdot^\circ$ is invariant, i.e. $M^\circ := \{x \in M \mid x^\circ = \{x\}\}$.

Ground Symbols. A type is *ground* if it contains no type variables. The set of ground types is denoted by GType. The type substitutions that map all type variables to ground types are written $\mathsf{GTSubst}$. A constant instance $c_\sigma \in \mathsf{CInst}$ is *ground* if its type σ is ground, and the set of ground constant instances is GCInst.

3.2 Definitional Theories

Definitional theories are theories that consist of definitions $u \equiv t$, with the constant instance or type u that is being defined on the left-hand side, and the defining term t on the right-hand side. Constant instances are defined by a term, and types are defined by a predicate. In short we say *symbol* for a constant instance or a type. Definitions of symbols can be of two kinds:

– A *constant instance definition* is of the form $c_\sigma \equiv t_\sigma$ where $c_\sigma \in \mathsf{CInst}^\circ$, $t \in \mathsf{Term}$ contains no free term variables, $\mathrm{TV}(t) \subseteq \mathrm{TV}(\sigma)$, and $c \notin \{\mathsf{rep}, \mathsf{abs}\}$.
– A *type definition* is of the form $\tau \equiv t_{\sigma \to \mathsf{bool}}$ where $\tau = (\alpha_1, \ldots, \alpha_{\mathrm{arOf}(k)})k \in \mathsf{Type}^\bullet$, $\alpha_i \in \mathsf{TVar}$ (for $0 \leq i \leq \mathrm{arOf}(k)$) distinct, $t \in \mathsf{Term}$ contains no free term variables, and $\mathrm{TV}(t) \subseteq \mathrm{TV}(\sigma)$.

We now define the semantics of constant and type definitions. A constant instance definition equates the constant instance with the defining term. A type definition asserts the existence of a bijection $\mathsf{rep}_{\tau \to \sigma}$ (with inverse $\mathsf{abs}_{\sigma \to \tau}$) between the type τ and the subset of σ that is given by the predicate t, provided this subset is non-empty. As a technical detail, as opposed to existentially quantifying over rep and abs in the resulting axiom [12], we assume that these two constants are present in the signature (with type $\alpha \to \beta$), and reserve them as not definable.

– A constant instance definition $c_\sigma \equiv t$ stands for the formula $c_\sigma \doteq t$.
– A type definition $\tau \equiv t_{\sigma \to \mathsf{bool}}$ stands for the formula

$$\begin{aligned}(\exists x_\sigma.\ t\ x) \quad \Rightarrow \quad & (\forall x_\sigma.\ t\ x \Rightarrow \mathsf{rep}_{\tau \to \sigma}\ (\mathsf{abs}_{\sigma \to \tau}\ x) \doteq x) \\ & \wedge (\forall y_\tau.\ t\ (\mathsf{rep}\ y)) \wedge (\forall y_\tau.\ \mathsf{abs}\ (\mathsf{rep}\ y) \doteq y)\end{aligned}$$

The signature needs to contain all logic symbols from these axioms (Sect. 3.3).

We impose two additional constraints [1,12] that ensure consistency of theories of definitions: orthogonality of definitions, and termination of a certain relation.

Orthogonality. Two types σ and τ are *orthogonal*, written $\sigma \mathbin{\#} \tau$, if they have no common instance, i.e. if for all $\rho, \rho' \in \mathsf{TSubst}$ $\rho(\sigma) \neq \rho'(\tau)$. Orthogonality extends to constant instances, written $c_\sigma \mathbin{\#} d_\tau$, if $c \neq d$ or $\sigma \mathbin{\#} \tau$. Two definitions $u \equiv t$, $v \equiv s$ are orthogonal if they are either of different kinds or if $u \mathbin{\#} v$.

Dependency Relation. Given a set of definitions D, the *dependency relation* for D, written $\rightsquigarrow_D$, is a binary relation on symbols. It tracks dependencies of defined symbols on their definiens, and is defined by $u \rightsquigarrow_D v$ (for $u, v \in \mathsf{CInst} \cup \mathsf{Type}$) if:

1. $u \equiv t \in D$ and $v \in t^\bullet \cup t^\circ$, or
2. u is a constant of type σ and $v \in \sigma^\bullet$, or
3. u is a type $(\alpha_1, \ldots, \alpha_{\mathrm{arOf}(k)})\, k$ and $v \in \{\alpha_1, \ldots, \alpha_{\mathrm{arOf}(k)}\}$.

We may simply write $\rightsquigarrow$ when D is clear from the context.

Type-Substitutive Closure. When R is a binary relation on $\mathsf{CInst} \cup \mathsf{Type}$, we write $R^{\downarrow}$ for the *type-substitutive closure* of R, defined as the smallest relation such that, for any $(s, t) \in R$ and any $\rho \in \mathsf{TSubst}$, $(\rho(s), \rho(t)) \in R^{\downarrow}$. Thus, the type-substitutive closure extends R to its image under arbitrary type substitutions.

Definitional Theories. A binary relation R is *terminating* (also converse well-founded or Noetherian) if there exists no infinite sequence $(a_i)_{i \in \mathbb{N}}$ such that $a_i \ R \ a_{i+1}$ for all $i \in \mathbb{N}$.

Finally, a theory D is *definitional* if it is a finite set of pairwise orthogonal definitions for which the relation $\rightsquigarrow_D{}^{\downarrow}$ is terminating. As a result of [1,12], every definitional theory has a (lazy ground semantics) model.

3.3 The Deductive System

We follow Kunčar and Popescu [10,11] in our description of the deductive system for HOL. The system is motivated by and abstracts from the implementation of higher-order logic in Isabelle.

We assume that the usual logical connectives and quantifiers are present in the signature, and define the set Ax of axioms to contain the formulae in Fig. 1. We note that the formulae on the left-hand side of the figure can be regarded as definitions of the logical connectives. For simplicity, we do not consider the axioms mem_Collect_eq and Collect_mem_eq in this paper, which introduce set-builder notation. The notation $t \not\doteq t'$ is shorthand for $\neg(t \doteq t')$. We let φ range over formulae.

Inference rules of the deductive system, given in Fig. 2, have the shape

$$\frac{T, \Gamma_1 \vdash \varphi_1 \quad \cdots \quad T, \Gamma_n \vdash \varphi_n}{T, \Gamma \vdash \varphi}$$

where T is the theory from which φ is derived by assuming the formulae in Γ [14]. Hereinafter, we deliberately omit empty assumption lists.

$\mathsf{True} \doteq ((\lambda x_{\mathsf{bool}}.\ x) \doteq (\lambda x_{\mathsf{bool}}.\ x))$	$\mathsf{refl}\colon x_\alpha \doteq x$
$\forall \doteq (\lambda p_{\alpha \to \mathsf{bool}}.\ (p \doteq (\lambda x.\ \mathsf{True})))$	$\mathsf{subst}\colon x_\alpha \doteq y \Rightarrow P\ x \Rightarrow P\ y$
$\exists \doteq (\lambda p_{\alpha \to \mathsf{bool}}.\ \forall\ (\lambda q.\ (\forall\ (\lambda x.\ p\ x \Rightarrow q)) \Rightarrow q))$	$\mathsf{iff}\colon (p \Rightarrow q) \Rightarrow (q \Rightarrow p) \Rightarrow (p \doteq q)$
$\mathsf{False} \doteq (\forall\ (\lambda p_{\mathsf{bool}}.\ p_{\mathsf{bool}}))$	$\mathsf{True\_or\_False}\colon (b \doteq \mathsf{True}) \vee (b \doteq \mathsf{False})$
$\neg \doteq (\lambda p.\ (p \Rightarrow \mathsf{False}))$	$\mathsf{some\_intro}_\alpha\colon p_{\alpha \to \mathsf{bool}}\ x \Rightarrow p\ (\mathsf{some}\ p)$
$\wedge \doteq (\lambda p\ q.\ \forall\ (\lambda r.\ (p \Rightarrow (q \Rightarrow r)) \Rightarrow r))$	$\mathsf{suc\_inj}\colon \mathsf{succ}\ x \doteq \mathsf{succ}\ y \Rightarrow x \doteq y$
$\vee \doteq (\lambda p\ q.\ \forall\ (\lambda r.\ (p \Rightarrow r) \Rightarrow (q \Rightarrow r) \Rightarrow r))$	$\mathsf{suc\_not\_zero}\colon\ \mathsf{succ}\ x \not\doteq \mathsf{zero}$

Fig. 1. Axioms for Isabelle/HOL.

$$\frac{}{T, \emptyset \vdash \varphi}\ [\varphi \in \mathsf{Ax} \cup T]\ \ (\textsc{Fact}) \qquad \frac{T, \Gamma \vdash f\ x_\sigma \doteq g\ x_\sigma}{T, \Gamma \vdash f \doteq g}\ [x_\sigma \notin \Gamma]\ \ (\textsc{Ext})$$

$$\frac{}{T, \Gamma \vdash (\lambda x_\sigma.\, t)\ s \doteq t[s/x_\sigma]}\ \ (\textsc{Beta}) \qquad \frac{T, \Gamma \vdash \psi}{T, \Gamma \setminus \{\varphi\} \vdash \varphi \Rightarrow \psi}\ \ (\textsc{ImpI})$$

$$\frac{}{T, \{\varphi\} \vdash \varphi}\ \ (\textsc{Assum}) \qquad \frac{T, \Gamma \vdash \varphi}{T, \Gamma \vdash (\varphi[\overline{\sigma}/\overline{\alpha}])[\overline{t}/\overline{x_\tau}]}\ [\overline{\alpha}, \overline{x_\tau} \notin \Gamma, t_i : \tau_i]\ \ (\textsc{Inst})$$

$$\frac{T, \Gamma_1 \vdash \varphi \Rightarrow \psi \quad T, \Gamma_2 \vdash \varphi}{T, \Gamma_1 \cup \Gamma_2 \vdash \psi}\ \ (\text{MP})$$

Fig. 2. Inference rules for Isabelle/HOL.

4 General Semantics

In this section, we define a general (Henkin-style) semantics for HOL. Our semantics generalises the lazy ground semantics of [1], by incorporating ideas of Henkin [2,7]: in a general model, function types may be interpreted by a proper (non-empty) subset of the corresponding function space.

Built-in Closure. For a set of types $T \subseteq \mathsf{Type}$ let $\mathsf{Cl}(T)$ denote the *built-in closure of* T, defined as the smallest set such that: $T \subseteq \mathsf{Cl}(T)$, $\mathsf{bool}, \mathsf{ind} \in \mathsf{Cl}(T)$, and for any two types $\sigma, \tau \in \mathsf{Cl}(T)$ the function type $\sigma \to \tau$ is in $\mathsf{Cl}(T)$. Thus, $\mathsf{Cl}(T)$ contains those types that can be constructed from T by repeated application of built-in type constructors.

Fragments. A *(signature) fragment* F is a pair $F = (T, C)$ of types $T \subseteq \mathsf{GType}^\bullet$ and constant instances $C \subseteq \mathsf{GInst}^\circ$, with the constraint that for each $c_\sigma \in C$: $\sigma \in \mathsf{Cl}(T)$. The types generated by the fragment are $\mathsf{Type}^F := \mathsf{Cl}(T)$, and its terms $\mathsf{Term}^F := \{t \in \mathsf{Term} \mid t^\bullet \subseteq \mathsf{Type}^F, t^\circ \subseteq C\}$ are those whose non-built-in symbols are from the fragment. Term^F is closed under taking sub-terms. We call the largest fragment $(\mathsf{GType}^\bullet, \mathsf{GInst}^\circ)$ the *total fragment.*

Fragment Pre-interpretations. We fix the set of Booleans $\mathbb{B} := \{\mathsf{true}, \mathsf{false}\}$. For a fragment $F = (T, C)$ an F*-pre-interpretation* is a pair of families $\mathcal{I} = \big(([\sigma])_{\sigma \in \mathsf{Type}^F}, ([c_\sigma])_{c_\sigma \in C}\big)$ that for all symbols $\sigma, \tau \in \mathsf{Type}^F$ and $c_\sigma \in C$ satisfies[1]

$$[\sigma] \neq \emptyset, \qquad [\mathsf{bool}] = \mathbb{B}, \qquad [\mathsf{ind}] = \mathbb{N},^{1} \qquad [c_\sigma] \in [\sigma], \text{ and } \qquad [\sigma \to \tau] \subseteq [\sigma] \to [\tau].$$

Valuations. A *valuation* for $\mathcal{I}$ is a function ξ_ρ parameterised by a ground type substitution $\rho \in \mathsf{GTSubst}$ that assigns meaning to all variables whose type is contained in the fragment, i.e. $\rho(\sigma) \in \mathsf{Type}^F$ implies $\xi_\rho(x_\sigma) \in [\rho(\sigma)]$ for all term variables $x_\sigma \in \mathsf{Var} \times \mathsf{Type}$.

[1] We use $[\mathsf{ind}] = \mathbb{N}$ for simplicity but could allow any infinite set. [1,12].

General Fragment Interpretations. For an F-pre-interpretation $\mathcal{I}$ and a valuation ξ_ρ, we define a partial function $[\![\cdot]\!]_{\xi_\rho}$ that extends $\mathcal{I}$ to terms t with $\rho(t) \in \mathsf{Term}^F$:

$$\begin{aligned} [\![x_\sigma]\!]_{\xi_\rho} &:= \xi_\rho(x_\sigma) & [\![c_\sigma]\!]_{\xi_\rho} &:= [c_{\rho(\sigma)}] \text{ for } c \text{ non-built-in} \\ [\![s\ t]\!]_{\xi_\rho} &:= [\![s]\!]_{\xi_\rho}([\![t]\!]_{\xi_\rho}) & [\![\lambda x_\sigma .\, t_\tau]\!]_{\xi_\rho} &: [\rho(\sigma)] \to [\rho(\tau)], z \mapsto [\![t_\tau]\!]_{\xi_\rho (\!| x \mapsto z |\!)} \end{aligned}$$

Here, function update $f(\!| x \mapsto z |\!)$ denotes the function that is equal to f except for the argument x, for which its value is z. The built-in constants have a fixed interpretation:

$$\begin{aligned} &[\![\Rightarrow_{\mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}}]\!]_{\xi_\rho} \text{ is implication} && [\![\mathsf{zero}_{\mathsf{ind}}]\!]_{\xi_\rho} := 0 \\ &[\![\dot{=}_{\sigma \to \sigma \to \mathsf{bool}}]\!]_{\xi_\rho} \text{ is equality on } [\rho(\sigma)] && [\![\mathsf{succ}_{\mathsf{ind} \to \mathsf{ind}}]\!]_{\xi_\rho} \text{ is the successor function} \\ &[\![\mathsf{some}_{(\sigma \to \mathsf{bool}) \to \sigma}]\!]_{\xi_\rho} \text{ is Hilbert-choice} \end{aligned}$$

In general, $[\![\cdot]\!]_{\xi_\rho}$ is partial because the interpretations of function types in $\mathcal{I}$ may not contain enough elements to interpret all terms. We say that $\mathcal{I}$ is a *general F-interpretation* if $[\![\cdot]\!]_{\xi_\rho}$ is total for every $\rho \in \mathsf{GTSubst}$ and every valuation ξ_ρ for $\mathcal{I}$, and $[\![t]\!]_{\xi_\rho} \in [\mathrm{tpOf}(\rho(t))]$ for all t with $\rho(t) \in \mathsf{Term}^F$.

General Models and Validity. A *general model* is a general $(\mathsf{GType}^\bullet, \mathsf{GCInst}^\circ)$-interpretation, i.e. a general interpretation for the total fragment. A general model is *standard* if $[\sigma \to \tau] = [\sigma] \to [\tau]$ for all σ, $\tau \in \mathsf{GType}$.

A formula φ is *valid* in a general model $\mathcal{M}$, written $\mathcal{M} \models \varphi$, if for all ground type substitutions $\rho \in \mathsf{GTSubst}$ and all valuations ξ_ρ it holds that $[\![\varphi]\!]_{\xi_\rho} = \mathsf{true}$. We write $\mathcal{M} \models E$ if all formulae within a set E are valid in a general model $\mathcal{M}$.

5 Results

In this section we derive model-theoretic conservativity, and prove soundness and completeness for the deductive system w.r.t. general semantics. We finish the section by transferring the semantic conservativity result to a syntactic one.

5.1 Model-Theoretic Conservativity

We discuss how symbols in a definitional theory extended by a symbol definition can be interpreted before and after the extension.

We write $\rightsquigarrow^{\downarrow *}$ for the reflexive-transitive type-substitutive closure of the dependency relation.

For a set of symbols U, we recall the *U-independent fragment F_U* [5,6]. In [6] F_U was introduced for singleton sets U (corresponding to the definition of a single symbol); this was generalised to arbitrary sets in [5].

Lemma 1. *Let D be a definitional theory, and let $U \subseteq \mathsf{Type}^\bullet \cup \mathsf{CInst}^\circ$. We write V_U for the pre-image of type instances of elements in U under the reflexive-transitive, type-substitutive closure of the dependency relation $\rightsquigarrow_D$, i.e.*

$$V_U := \left\{ v \in \mathsf{GType}^\bullet \cup \mathsf{GCInst}^\circ \mid \exists u \in U, \rho \in \mathsf{TSubst}.\ v\ ({\rightsquigarrow_D}^{\downarrow})^*\ \rho(u) \right\}.$$

*Then $F_U := (\mathsf{GType}^\bullet \setminus V_U, \mathsf{GCInst}^\circ \setminus V_U)$ is a fragment, called the U-*independent *fragment.*

Model-theoretic conservativity as we prove it extends a model for possibly several definitional updates, keeping the interpretations for all symbols that are independent of the updates, i.e. all symbols that are in the independent fragment where U are the updated symbols.

In the proof a part of a general model $\mathcal{M}$ of a definitional theory D is expanded to a general model of a definitional extension D' by well-founded recursion over parts of the $\rightsquigarrow^{\downarrow}$ relation. The part of the general model that is extended corresponds to the fragment independent of the defined symbols from $D' \setminus D$. The earlier monolithic model construction [1,12] obtains a model from the ground up, by recursion over the entire $\rightsquigarrow^{\downarrow}$ relation. The incremental model construction in [5,6] considers standard models and extension by a single definition (although the latter can be generalised to any finite definitional extension).

We make use of a fragment of dependencies $E_{\rightsquigarrow_D}$, which is defined as all instances of elements in E and their dependencies.

$$E_{\rightsquigarrow_D} := \left\{ x \in \mathsf{GType}^\bullet \cup \mathsf{GCInst}^\circ \middle| \exists u \in E, \rho \in \mathsf{TSubst} : \rho(u) {\rightsquigarrow_D}^{\downarrow *} x \right\}$$

Indeed, this is a fragment: for any $c_\sigma \in E_{\rightsquigarrow_D}$ we have $\sigma \in \mathsf{Cl}(\mathsf{Type} \cap E_{\rightsquigarrow_D})$, which is simply by $c_\sigma \rightsquigarrow_D v$ for $v \in \sigma^\bullet$ because $\sigma \in \mathsf{Cl}(\sigma^\bullet)$.

Theorem 1 (Model-theoretic conservativity). *Let $\mathcal{M}$ be a general model of a definitional theory D, and let $D' \supseteq D$ be a definitional extension of D (by possibly several definitions). Let U be the symbols defined in $D' \setminus D$, i.e. $U = \{u \mid \exists t.\, u \equiv t \in D' \setminus D\}$. There exists a general model $\mathcal{M}'$ of the extended theory D' with the following property: the models $\mathcal{M}$ and $\mathcal{M}'$ agree on the interpretation of all types and terms in $\mathsf{Type}^{F_U} \cup \mathsf{Term}^{F_U}$.*

Proof. Any dependency relation in this proof is w.r.t. D'. Let U be the set of symbols defined in $D' \setminus D$ and let V_U be defined as in Lemma 1. We define an interpretation for the total fragment $\top = (\mathsf{GType}^\bullet, \mathsf{GCInst}^\circ)$. For any ground symbol $w \in \top \setminus V_U$ we define $[w]$ as the interpretation of w within the model $\mathcal{M}$. For all elements in V_U we define an interpretation by well-founded recursion over $\rightsquigarrow^{\downarrow +}$, which is the transitive closure of the type substitutive dependency relation. From the definitional theory D' we get a terminating dependency relation $\rightsquigarrow^{\downarrow +}$ [1], and we can define an interpretation $[v]$ for any symbol $v \in V_U$, basing on $[w]$ for w such that $v \rightsquigarrow^{\downarrow +} w$, and their standard interpretation. Thus in each step we define the interpretation for v from the interpretation of symbols $E_{\rightsquigarrow}$ for $E = \{w | v \rightsquigarrow^{\downarrow +} w\} \cup F_U$.

We say $v \in \mathsf{GSymb}$ *matches* definition $(s \equiv r) \in D'$ if $v = \rho(s)$ is a type instance of s for $\rho \in \mathsf{GTSubst}$. Due to orthogonality of D', a matching definition is uniquely determined. If there is no match for v, as v is ground, it is orthogonal to any definition in D'.

Case 1. v matches $s \equiv r$ and $v = \rho(s)$. Because $v \rightsquigarrow^{\downarrow +} \rho(r)^{\bullet}$ and $v \rightsquigarrow^{\downarrow +} \rho(r)^{\circ}$ the closed term $\rho(r)$ has an interpretation $[\![r]\!]_{\xi_\rho}$ for an arbitrary ξ_ρ w.r.t. ρ.

Sub-case 1.1. s is a constant instance. We assign $[v] = [\![r]\!]_{\xi_\rho}$.

Sub-case 1.2. s is a (non-built-in) type with $\mathrm{tpOf}(r) = \sigma \to \mathsf{bool}$. We set $[v] = \big\{a \in [\rho(\sigma)] \big| [\![r]\!]_{\xi_\rho}(a) = \mathsf{true}\big\}$ if this set is non-empty and $[v] = \mathbf{1}$ alternatively.

Case 2. there is no matching definition for v in D'. We distinguish two sub-cases.

Sub-case 2.1. v is a non-built-in type. If $v = \tau \to \sigma$ we define $[v] = [\tau] \to [\sigma]$. Otherwise we define $[v] = \mathbf{1}$.

Sub-case 2.2. v is a constant instance and $v \rightsquigarrow^{\downarrow +} \mathrm{tpOf}(v)^{\bullet}$.

- $v = \mathsf{abs}_{\sigma \to \tau}$, or $v = \mathsf{rep}_{\tau \to \sigma}$ and there is a definition in $s \equiv r \in D'$ with a type substitution $\rho \in \mathsf{TSubst}$ such that $\rho(s) = \sigma$ and $\rho(\mathrm{tpOf}(r)) = \tau \to \mathsf{bool}$. (Again, due to orthogonality there can only be one such matching definition.) In this case $[v]$ is undefined and it holds $[\sigma] \subseteq [\tau]$ as $\sigma \in V_u$.
  - $v = \mathsf{abs}_{\sigma \to \tau}$ then we define $[v]$ as the (identical) embedding $[\sigma] \subseteq [\tau]$ which exists in $[\sigma \to \tau]$, or
  - $v = \mathsf{rep}_{\tau \to \sigma}$ then we define $[v]$ as some embedding of $[\tau]$ to $[\sigma]$ such that $[v]\big|_{[\sigma]}$ is the identity
- $v = c_\sigma$ for some $c_\sigma \in \mathsf{GCInst}^{\circ}$ then we define $[v] = \mathsf{choice}([\sigma])$.

We get a model from the constructed pre-interpretation $[\,\cdot\,]$ of $\top$ by $[\![\,\cdot\,]\!]$ to arrive at an interpretation of terms from Term.

We prove that the recursively constructed total fragment interpretation is indeed a model for D'. For types $\tau \in \mathsf{Type}^{E_\rightsquigarrow}$ the built-in constants equality $\doteq_{\tau \to \tau \to \mathsf{bool}}$ and Hilbert-choice $\mathsf{some}_{(\tau \to \mathsf{bool}) \to \tau}$ are both interpretable within any of their ground types either by induction hypothesis, or otherwise as both of the constants' types are the full function spaces.

Let $s \equiv r \in D'$, and let $\rho \in \mathsf{GTSubst}$ be a ground type substitution for which we show $[s \equiv r]_{\xi_\rho} = \mathsf{true}$ for an arbitrary ξ_ρ (as $s \equiv r$ is a closed term). By orthogonality, $\rho(s)$ matches only the definition $s \equiv r$. If $\rho(s) \notin V_U$ then $\rho(s \equiv r) \in \mathsf{Term}^{F_U}$, and $s \equiv r \in D$, and for (even arbitrary) ρ and any ξ_ρ it holds, with the interpretation from $\mathcal{M}$ written as $\mathcal{M}(\cdot)$:

$$[s \equiv r]_{\xi_\rho} = \mathcal{M}(s \equiv r)_{\xi_\rho} = \mathsf{true}.$$

Otherwise if $\rho(s) \in V_U$ we distinguish by the kind of s. If s is a constant instance by the first case

$$[s \equiv r]_{\xi_\rho} = [\![s \doteq r]\!]_{\xi_\rho} = [\![s]\!]_{\xi_\rho}[\![\doteq]\!]_{\xi_\rho}[\![r]\!]_{\xi_\rho} = [\![r]\!]_{\xi_\rho}[\![\doteq]\!]_{\xi_\rho}[\![r]\!]_{\xi_\rho} = \mathsf{true}$$

for any ξ_ρ. If $s = \tau$ is a type instance and the predicate $[\![r_{\sigma\to\mathsf{bool}}]\!]_{\xi_\rho}$ is satisfied for some element of $[\rho(\sigma)]$, proving $[\![s \equiv r]\!]_{\xi_\rho} = \mathsf{true}$ means to prove that the properties for $\mathsf{rep}_{\rho(\sigma\to\sigma)}$ and $\mathsf{abs}_{\rho(\tau\to\sigma)}$ hold, because the first-order logic operators are behaving in a standard sense. For any $a \in [\rho(\sigma)]$ by definition we have $[\![r]\!]_{\xi_\rho}(a)$ which is the first conjunct as $[\mathsf{rep}_{\rho(\sigma\to\tau)}]([\rho(\sigma)]) = [\rho(\sigma)] \subseteq [\rho(\tau)]$. Similarly by sub-case 2.2 above, both

$$[\mathsf{rep}_{\rho(\tau\to\sigma)}] \circ [\mathsf{abs}_{\rho(\sigma\to\tau)}] \qquad \text{and} \qquad [\mathsf{abs}_{\rho(\sigma\to\tau)}] \circ [\mathsf{rep}_{\rho(\tau\to\sigma)}]$$

are identity on $[\rho(\tau)] \subseteq [\rho(\sigma)]$.

The axioms trivially hold, also by the definition of general model. □

In the proof a model for general semantics can be extended, as terms from the fragment of all dependencies $E_{\rightsquigarrow_D}$ are interpretable and as function types within the recursion are interpreted by the full set-theoretic function space. For example, for a type τ whose interpretation is defined in the recursion, the type $\tau \to \mathsf{bool}$ of predicates on τ is interpreted by the set of all functions $[\tau] \to \mathbb{B}$.

Corollary 1. *Any definitional theory has a general model.*

The proof is by induction on the size of the theory. The base case is the empty theory, which has a (standard) model. The induction step is by Theorem 1. (Of course, the corollary also follows directly from the stronger result that any definitional theory has a standard model [1,12].)

5.2 Soundness

The deductive system is sound w.r.t. general semantics: any formula that is derivable is valid in all general models.

Theorem 2 (Soundness for general semantics). *Let T be a set of formulae. If φ is a formula such that $T \vdash \varphi$, then for every general model $\mathcal{M}$ with $\mathcal{M} \models T$ it holds that $\mathcal{M} \models \varphi$.*

The proof is by induction over the derivation $T \vdash \varphi$ (which we omit here). Since any definitional theory has a general model, soundness implies that False is not derivable, i.e. any definitional theory is (syntactically) *consistent*.

5.3 Completeness

In this section we lay out the completeness proof of the deductive system with respect to general models. Completeness means for any formula which is valid in all models of a theory there is a proof from the theory. Our proof follows the argumentation of Henkin [2,7]: we show that any consistent theory is contained in a consistent super-set that is *extensionally complete*. From these properties we construct a general model, which is also a model of the original theory. Completeness follows from the existence of a model for any consistent set in Theorem 4.

Negation-Complete. A theory T is *negation-complete* if for any closed formula φ, we can derive $T \vdash \varphi$ or $T \vdash \neg\varphi$.

Extensionally Complete. A theory T is *extensionally complete* if for any closed term t and t' there is an a such that $T \vdash (t\ a \doteq t'\ a) \Rightarrow (t \doteq t')$. The contrapositive $t \not\doteq t' \Rightarrow t\ a \not\doteq t'\ a$ means that a witnesses the inequality of t and t'.

Lemma 2 (Extension Lemma). *Any finite consistent set of formulae has a consistent, negation-complete and extensionally complete theory extension.*

Proof. For a finite theory T in the given countably infinite language we construct a theory extension T' which satisfies the stated properties.

We well-order all closed formulae of the language of T and denote them by ψ^n for n a natural number. Let $\mathcal{C}$ denote the set of countably many constants Const whose type is a type variable minus all (finitely many) constants that syntactically appear within T. We define a sequence $(T_n)_{n\in\mathbb{N}_0}$ of theory extensions of T.

For a theory T_n and for two closed terms $t_{\sigma\to\tau}$ and t' of same type, let a be the constant instance of type σ whose name is smallest in $\mathcal{C}$ less all constant names from T_n. The finite number of symbols used in T_n is at most n plus the number of symbols appearing in T. We define $T_0 := T$, $T' := \bigcup_{n\in\mathbb{N}_0} T_n$ and

$$T_{n+1} := \begin{cases} T_n \cup \{\psi^n\} & \text{if the union is consistent} \\ T_n & \text{otherwise, if } \psi^n \text{ is no equality of two functions} \\ T_n \cup \{t\ a \ \not\doteq t'\ a\} & \text{otherwise, for } \psi^n = (t \doteq t') \text{ and } a \text{ as described} \end{cases}$$

By induction any T_n is consistent: Assuming T_n is consistent, we prove its successor T_{n+1} consistent. For the first two sub-cases consistency is immediate. For the third sub-case, assume that a is chosen as remarked above and $T_n \cup \{t \doteq t'\}$ is inconsistent. Consequently, $T_n \vdash t \not\doteq t'$. Suppose that also $T_n \cup \{t\ a \not\doteq t'\ a\}$ is inconsistent, and thus we have $T_n \vdash t\ a \doteq t'\ a$. It is derivable that $t \doteq t'$ is equivalent to $t\,x \doteq t'\,x$. Instantiating the contrapositive at a, we get $T_n \vdash t\ a \not\doteq t'\ a$ in contradiction to consistency of T_n.

We show that T' is negation-complete. Let φ be any closed formula. Then there is an n such that $\psi^n = \varphi$. If $T_{n+1} = T_n \cup \{\varphi\}$ then $T' \vdash \varphi$. Otherwise $T_n \cup \{\varphi\}$ is inconsistent and by law of excluded middle we derive $T' \vdash \neg\varphi$.

We prove that T' is extensionally complete. For two closed terms t and t' of same function type, if $T' \vdash t \doteq t'$ then by extensionality this holds at any x: $T' \vdash t\,x \doteq t'\,x$. Otherwise there is n such that $\psi^n = t \doteq t'$ and T_{n+1} is defined as $T_n \cup \{t\,a \not\doteq t'\,a\}$ for some a. $\square$

We will use the previous Lemma 2 to obtain an extensionally complete extension of a definitional theory. This extension T' is not a definitional theory, but this is not problematic.

For a theory we want consistency to be equivalent to the existence of a general model. To construct a model of a consistent theory, we—different from Henkin and Andrews—also define interpretations for (non-built in) types. We show that the construction gives a model for our semantics.

Theorem 3 (Henkin's Theorem). *Every consistent set of closed formulae has a general model.*

Proof. Let T' be an extension according to Lemma 2 of a consistent set of formula. We define an interpretation $[t]_{t\in\mathsf{Term}}$ (initially only for closed ground terms) and $[\sigma]_{\sigma\in\mathsf{GType}}$ for the extension T'. The construction goes by induction on the types $\sigma \in \mathsf{GType}$ such that the two properties hold:

1. $[\sigma] = \{[t_\sigma] | t_\sigma$ closed (ground) term of type $\sigma\}$
2. for all closed (ground) terms s_σ, t_σ: $[t_\sigma] = [s_\sigma]$ iff $T' \vdash t_\sigma \doteq s_\sigma$

At several instances we use that ground types are closed under the sub-type relation. Closed terms of type σ, like $\mathsf{some}(\lambda x_\sigma.\mathsf{True})$, ensure that each type's interpretation $[\sigma]$ is non-empty if defined according to Item 1.

Booleans. For $[\mathsf{bool}] := \mathbb{B}$, Items 1 and 2 hold, when setting $[\varphi_{\mathsf{bool}}] := \mathsf{true}$ iff $T' \vdash \varphi_{\mathsf{bool}}$ and otherwise $[\varphi_{\mathsf{bool}}] := \mathsf{false}$ as by maximality of T' otherwise the formula $\neg\varphi_{\mathsf{bool}}$ is deducible from T'.

Natural Numbers. Define $[\mathsf{ind}] := \mathbb{N}$. Define for t_{ind} a closed term of type ind:

$$[t_{\mathsf{ind}}] := \{n \in \mathbb{N} | T' \vdash t_{\mathsf{ind}} \doteq \underbrace{\mathsf{succ}(\ldots(\mathsf{succ}}_{n \text{ times}}(\mathsf{zero}))\ldots)\}$$

We have $[\mathsf{ind}] = \{[t_{\mathsf{ind}}] | t_{\mathsf{ind}}$ a closed term$\}$, as reflexivity of $\doteq$ gives $(\subseteq)$ and on the other hand the only possible constructors for the ind type are zero and succ. Clearly both properties hold.

Function Types. Let $\sigma, \tau \in \mathsf{GType}$, $[\sigma]$ and $[\tau]$ be defined. We define

$$[t_{\sigma\to\tau}] : [\sigma] \to [\tau], \ \ [s_\sigma] \mapsto [t_{\sigma\to\tau}\ s_\sigma].$$

The term $[t_{\sigma\to\tau}][s_\sigma]$ is well-defined, as the choice of s_σ as the representative of the equivalence class $[s_\sigma]$ is irrelevant: let s'_σ be such that $[s_\sigma] = [s'_\sigma]$ holds and thus equivalently $T' \vdash s_\sigma \doteq s'_\sigma$, which implies $T' \vdash t_{\sigma\to\tau}\ s_\sigma \doteq t_{\sigma\to\tau}\ s'_\sigma$.

We define $[\sigma \to \tau] := \{[t_{\sigma\to\tau}] | t_{\sigma\to\tau}$ a closed term$\}$ satisfying Item 1 and proceed with the proof of Item 2. Let $t_{\sigma\to\tau}$, $t'_{\sigma\to\tau}$ and s_σ be closed terms. If $T' \vdash t_{\sigma\to\tau} \doteq t'_{\sigma\to\tau}$ then especially $t_{\sigma\to\tau}\ s_\sigma \doteq t'_{\sigma\to\tau}\ s_\sigma$ is provable in T'. By

$$[t_{\sigma\to\tau}][s_\sigma] = [t_{\sigma\to\tau}\ s_\sigma] = [t'_{\sigma\to\tau}\ s_\sigma] = [t'_{\sigma\to\tau}][s_\sigma],$$

the functions $[t_{\sigma\to\tau}]$ and $[t'_{\sigma\to\tau}]$ coincide at every value.

On the other hand assume the equality $[t_{\sigma\to\tau}] = [t'_{\sigma\to\tau}]$. As T' is extensionally complete there exists a closed term s_σ for the term $t_{\sigma\to\tau} \doteq t'_{\sigma\to\tau}$ such that

$$(t_{\sigma\to\tau} s_\sigma \doteq t'_{\sigma\to\tau} s_\sigma) \Rightarrow (t_{\sigma\to\tau} \doteq t'_{\sigma\to\tau})$$

holds in T'. By $[t_{\sigma\to\tau}\ s_\sigma] = [t_{\sigma\to\tau}][s_\sigma] = [t'_{\sigma\to\tau}][s_\sigma] = [t'_{\sigma\to\tau}\ s_\sigma]$, and induction hypothesis, we have that $t_{\sigma\to\tau} s_\sigma \doteq t'_{\sigma\to\tau} s_\sigma$ holds in T'. Ultimately it holds:

$$[t_{\sigma\to\tau}\ s_\sigma] = [t'_{\sigma\to\tau}\ s_\sigma] \qquad \text{iff} \qquad T' \vdash t_{\sigma\to\tau}\ s_\sigma \doteq t'_{\sigma\to\tau}\ s_\sigma.$$

Non-built-in types. All other types $\sigma \in \mathsf{GType}$ are either instances of a definition or are undefined. In any case we define the interpretation as the equivalence class for all closed (ground) terms t_σ, which satisfies Items 1 and 2.

$$\begin{aligned}[t_\sigma] &= \{s_\sigma | T' \vdash t_\sigma \doteq s_\sigma \text{ for closed } s_\sigma\}\\ [\sigma] &= \{[t_\sigma] | t_\sigma \text{ closed (ground) term of type } \sigma\}\end{aligned}$$

If $T' \vdash \exists x_\sigma.\ t\ x_\sigma$ then we can derive from T' that

$$(\forall x_\sigma.\, t\, x \Rightarrow \mathsf{rep}_{\tau\to\sigma}\,(\mathsf{abs}_{\sigma\to\tau}\ x) \doteq x) \wedge \forall y_\tau.\, t\ (\mathsf{rep}\, y) \wedge \mathsf{abs}\,(\mathsf{rep}\, y) \doteq y$$

As the properties of quantifiers, equality and logical connectives proof-theoretically correspond to those of the meta logic, we get al.l the desired properties for the interpretations of rep and abs.

General Model. Now all ground closed terms are interpretable w.r.t. $[\cdot]$. For this to become a model we define a function $[\cdot]_{\xi_\rho}$ that will become the interpretation function for this model, to also interpret terms w.r.t. a ground type substitution ρ and a term variable assignment ξ_ρ.

We write y^{ξ_ρ} for t such that $[t] = \xi_\rho(y)$ and extend interpretation to non-ground formulae by defining $[t']_{\xi_\rho} = [\rho(t'[x^{\xi_\rho}/x]_{x\in\mathrm{FV}(t)})]_{\xi_\rho}$ for any ground type substitution ρ and any ξ_ρ such that $\xi_\rho(x_\sigma) \in [\rho(\sigma)]$ for any term variable x_σ. By induction we see that this gives an interpretation with the desired properties:

- $[x_\sigma]_{\xi_\rho} = [\rho(x^{\xi_\rho})] = [{x_\sigma}^{\xi_\rho}] = \xi_\rho(x_\sigma) \in [\rho(\sigma)]$
- $[c_\sigma]_{\xi_\rho} = [c_{\rho(\sigma)}]$
- Syntactic juggling gives $[s\,t]_{\xi_\rho} = \left[\rho\left((s\,t)\left[y^{\xi_\rho}/y\right]_{y\in\mathrm{FV}(s\,t)}\right)\right]$

$$= \left[\rho\left(s\left[y^{\xi_\rho}/y\right]_{y\in\mathrm{FV}(s)}\right)\right]\left[\rho\left(t\left[y^{\xi_\rho}/y\right]_{y\in\mathrm{FV}(t)}\right)\right] = [s]_{\xi_\rho}\,[t]_{\xi_\rho}.$$

- By Beta rule for any closed term a of type σ we have $T' \vdash (\lambda x_\sigma.\ t'_\tau)\ a \doteq t'_\tau[a/x_\sigma]$ for any t'_τ. Followed by type instantiation and meta-level rewriting (note that ρ is ground) we have that $T' \vdash \rho\,((\lambda x_\sigma.\ t'_\tau)\ a) \doteq \rho\,(t'_\tau[\rho(a)/x_\sigma])$. Let $[s]$ be an arbitrary element of $[\rho(\sigma)]$, then the above holds for $\rho(a) = s$

and for $t'_\tau = t_\tau \left[y^{\xi_\rho}/y\right]_{y \in \mathrm{FV}(\lambda x_\sigma.\, t_\tau)}$. And thus (without loss of generality rewriting variable names) we have

$$\begin{aligned}[\lambda x_\sigma.\, t_\tau]_{\xi_\rho}[s] &= \left[\rho\left((\lambda x_\sigma.\, t_\tau)\left[y^{\xi_\rho}/y\right]_{y \in \mathrm{FV}(\lambda x_\sigma.\, t_\tau)}\right)\right][s] \\ &= \left[\rho\left((\lambda x_\sigma.\, t_\tau)\left[y^{\xi_\rho}/y\right]_{y \in \mathrm{FV}(\lambda x_\sigma.\, t_\tau)}\, s\right)\right] \\ &= \left[\rho\left(t_\tau\left[y^{\xi_\rho(\!|x_\sigma \mapsto [s]|\!)}\big/ y\right]_{y \in \mathrm{FV}(t_\tau)}\right)\right] = [t_\tau]_{\xi_\rho(\!|x_\sigma \mapsto [s]|\!)}.\end{aligned}$$

For the built-in constant $\mathsf{some}_{(\sigma \to \mathsf{bool}) \to \sigma}$ we require that $[\mathsf{some}_{(\sigma \to \mathsf{bool}) \to \sigma}]$ is Hilbert-choice. For any $[p_{\sigma \to \mathsf{bool}}]$ and any $[x] \in [\sigma]$ that satisfies $[p]$, we show $[p\ (\mathsf{some}\ p)]$ holds. By $[p][x] = [p\ x] = \mathsf{true}$ we get $T' \vdash p\ x$ equivalently. From the axiom we prove $T' \vdash p\ (\mathsf{some}\ p) \doteq \mathsf{True}$, thus $[p\ (\mathsf{some}\ p)]$ holds.

Also for any ground type τ, equality $\doteq_{\tau \to \tau \to \mathsf{bool}}$ should be interpreted as equality on $[\tau]$. Immediately by construction, we have for any $[s], [t] \in [\tau]$ that

$$[s] = [t] \quad \text{iff} \quad T' \vdash s \doteq t \quad \text{iff} \quad [s][\doteq][t] = [s \doteq t] = \mathsf{true}.$$

As any ground type is non-empty, there are no empty types overall. Hence, as the axioms hold and as there is a valuation function this gives a general model. The constructed model is a general model of T' and any of its subsets. □

The semantics $[\![t]\!]_{\xi_\rho}$ of a term t is defined to apply type substitutions *lazily*, i.e. at the latest possible moment when interpreting a term (and never to term variables). Applying type substitutions eagerly would erroneously force equal valuation of distinct variables, e.g. x_α and x_{bool} under a type substitution ρ with $\rho(\alpha) = \mathsf{bool}$ [1].

However in the proof we see a different characterisation of the term semantics, namely as $[\![t]\!]_{\xi_\rho} = [\![\rho(t[y^{\xi_\rho}/y]_{y \in \mathrm{FV}(t)})]\!]_{\xi_\rho}$, which eagerly applies ρ to t, but only after replacing each free variable $y \in \mathrm{FV}(t)$. Therein y is replaced by a closed term y^{ξ_ρ} that has the same interpretation. The resulting term $\rho(t[y^{\xi_\rho}/y]_{y \in \mathrm{FV}(t)})$ is closed, hence a capture of term variables does not occur. This characterisation requires for any type σ and valuation ξ_ρ that the function $[\![\cdot]\!]_{\xi_\rho} : \mathsf{Term}_\sigma \to [\rho(\sigma)]$ is surjective, which is not given in Åman Pohjola and Gengelbach [1].

Completeness follows by Theorem 3 and a generic argument.

Theorem 4 (Completeness for general semantics). *Let T be a set of formulae. If φ is a formula that is valid in every general model $\mathcal{M}$ with $\mathcal{M} \models T$, then $T \vdash \varphi$.*

Proof. Let ψ be a universal closure of φ, then also ψ is valid in every general model of T. Suppose that $T' := T \cup \{\neg\psi\}$ is consistent, then by Theorem 3 let $\mathcal{M}$ be a general model of T'. We have $\mathcal{M} \models \neg\psi$ but also $\mathcal{M} \models T$, hence $\mathcal{M} \models \psi$. This is a contradiction. Consequently $T \cup \{\neg\psi\}$ is inconsistent, i.e. $T \cup \{\neg\psi\} \vdash \mathsf{False}$, thus $T \vdash \psi$. Eliminating the universal closure, we obtain $T \vdash \varphi$. □

As noted by a reviewer, completeness w.r.t. general lazy ground semantics can also be proved more directly from completeness of the monomorphic calculus. We briefly sketch the idea (but we have not worked out the technical details). If a (polymorphic) theory T is consistent, the theory of all ground instances of formulas in T is consistent. By completeness this theory has a (general) model. This model is also a model of T w.r.t. ground semantics.

5.4 Proof-Theoretic Conservativity

Having proven the deductive system with general semantics sound and complete, we derive proof-theoretic conservativity from model-theoretic conservativity in this section.

Lemma 3. *If a deductive system is sound and complete then model-theoretic conservativity implies proof-theoretic conservativity.*

Proof. Consider a theory extension $D' \supseteq D$. Assume $D' \vdash \varphi$ (with suitable assumptions on φ). We need to show $D \vdash \varphi$. Due to completeness it is sufficient to prove that φ holds in all models of D. By model-theoretic conservativity, for any model $\mathcal{M}$ of D there is a model $\mathcal{M}'$ of D' such that $\mathcal{M}'$ agrees with $\mathcal{M}$ on the interpretation of φ. Since $D' \vdash \varphi$, soundness implies that $\mathcal{M}'$ is a model of φ, hence $\mathcal{M}$ is a model of φ. □

We recall the main model-theoretic conservativity theorem [6, Theorem 3.3]. For U a set of symbol, the *U-independent fragment* F_U [5,6] defines the ground symbols which are not in the pre-image of type instances of U under the reflexive-transitive, type-substitutive closure of the dependency relation. The terms Term^{F_U} are all terms whose ground instances can be interpreted within F_U and accordingly are Type^{F_U} all the types that occur in the terms Term^{F_U} (see also Theorem 1).

Lemma 3 shows how the constraint in our model-theoretic notion of conservativity translates to a syntactic constraint:

Theorem 5 (Proof-theoretic conservativity). *Let D be a definitional theory and $D' \supseteq D$ be a definitional theory extension of D (by possibly several definitions), and let U be the set of symbols defined in $D' \setminus D$.*
If $\varphi_{\mathsf{bool}} \in \mathsf{Term}^{F_U}$ and $D' \vdash \varphi$ then $D \vdash \varphi$.

Proof. For a definitional theory extension $D' \supseteq D$, $\varphi_{\mathsf{bool}} \in \mathsf{Term}^{F_U}$ and $D' \vdash \varphi$, it is sufficient to prove that φ holds in all models of D, due to completeness (Theorem 4). Let $\mathcal{M}$ be a model of D. From model-theoretic conservativity (Theorem 1), we obtain a model extension $\mathcal{M}'$ for D' such that $\mathcal{M}$ and $\mathcal{M}'$ agree on the interpretation of all types and terms in $\mathsf{Type}^{F_U} \cup \mathsf{Term}^{F_U}$. Consequently, we have $\mathcal{M} \models \varphi$ iff $\mathcal{M}' \models \varphi$. Soundness implies that $\mathcal{M}' \models \varphi$, hence also $\mathcal{M} \models \varphi$.
□

6 Conclusion

We introduced general models as a generalisation of standard models, and proved soundness and completeness (by using ideas by Henkin [2,7]) for HOL with ad-hoc overloading. We extended our earlier model-theoretic conservativity result [5] to general models, and applied these results to show proof-theoretic conservativity for HOL with ad-hoc overloading: for a definitional theory extension $D' \supseteq D$, any formula φ that is derivable in D' and that does not (explicitly or implicitly) depend on symbols defined in $D' \setminus D$ is already derivable in D.

The established notion of proof-theoretic conservativity should be extensible to hold for related settings, for example with further axioms [14], with Arthan's constant specification [3], and with explicit signature extensions like in [1]. Using our results and unfolding definitions with ideas from [11] may allow a *relative* meta-safety result, i.e. unfolding the symbols defined in a definitional extension relative to an arbitrary definitional base theory.

Acknowledgements. We thank Andrei Popescu, Georg Struth and the anonymous reviewers for their valuable feedback on earlier versions of this paper.

References

1. Åman Pohjola, J., Gengelbach, A.: A mechanised semantics for HOL with ad-hoc overloading. In: Albert, E., Kovács, L. (eds.) LPAR23. LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning. EPiC Series in Computing, vol. 73, pp. 498–515. EasyChair (2020). https://doi.org/10.29007/413d, https://easychair.org/publications/paper/9Hcd
2. Andrews, P.B.: An Introduction to Mathematical Logic and Type Theory: To Truth through Proof. Applied logic series, No. 27, 2nd edn.. Kluwer Academic Publishers, Dordrecht; Boston (2002)
3. Arthan, R.: HOL constant definition done right. In: Klein, G., Gamboa, R. (eds.) ITP 2014. LNCS, vol. 8558, pp. 531–536. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08970-6_34
4. Church, A.: A formulation of the simple theory of types. J. Symbol. Logic **5**(02), 56–68 (1940). https://doi.org/10.2307/2266170, http://www.journals.cambridge.org/abstract_S0022481200108187
5. Gengelbach, A., Åman Pohjola, J., Weber, T.: Mechanisation of Model-theoretic Conservative Extension for HOL with Ad-hoc Overloading (2020, Under submission)
6. Gengelbach, A., Weber, T.: Model-theoretic conservative extension of definitional theories. In: Proceedings of 12th Workshop on Logical and Semantic Frameworks with Applications (LSFA 2017), pp. 4–16. Brasília, Brasil, September 2017. https://doi.org/10.1016/j.entcs.2018.10.009
7. Henkin, L.: Completeness in the theory of types. J. Symbol. Logic **15**(2), 81–91 (1950). https://doi.org/10.2307/2266967
8. Kuncar, O.: Correctness of Isabelle's cyclicity checker: implementability of overloading in proof assistants. In: Leroy, X., Tiu, A. (eds.) Proceedings of the 2015 Conference on Certified Programs and Proofs, CPP 2015, Mumbai, India, January 15–17, 2015, pp. 85–94. ACM (2015). https://doi.org/10.1145/2676724.2693175, https://doi.org/10.1145/2676724.2693175

9. Kunčar, O., Popescu, A.: Safety and conservativity of definitions in HOL and Isabelle/HOL. Proc. ACM Program. Lang. 2(POPL), 24:1–24:26 (2017). https://doi.org/10.1145/3158112
10. Kunčar, O.: Types, Abstraction and Parametric Polymorphism in Higher-Order Logic. Ph.D. thesis, Technische Universität München (2016). http://www21.in.tum.de/~kuncar/documents/kuncar-phdthesis.pdf
11. Kunčar, O., Popescu, A.: Comprehending Isabelle/HOL's consistency. In: Yang, H. (ed.) ESOP 2017. LNCS, vol. 10201, pp. 724–749. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54434-1_27
12. Kunčar, O., Popescu, A.: A consistent foundation for Isabelle/HOL. J. Autom. Reasoning **62**(4), 531–555 (2018). https://doi.org/10.1007/s10817-018-9454-8
13. Nipkow, T., Wenzel, M., Paulson, L.C. (eds.): Isabelle/HOL - A Proof Assistant for Higher-Order Logic. LNCS, vol. 2283. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45949-9
14. Norrish, M., Slind, K., et al.: The HOL System LOGIC, August 2019. http://sourceforge.net/projects/hol/files/hol/kananaskis-13/kananaskis-13-logic.pdf/download
15. Pitts, A.: The HOL logic. In: Introduction to HOL: A Theorem Proving Environment for Higher Order Logic, pp. 191–232. Cambridge University Press, Cambridge; New York (1993)
16. Wenzel, M.: Type classes and overloading in higher-order logic. In: Gunter, E.L., Felty, A. (eds.) TPHOLs 1997. LNCS, vol. 1275, pp. 307–322. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0028402