SUPERPOSITION FOR LAMBDA-FREE HIGHER-ORDER LOGIC

ALEXANDER BENTKAMP, JASMIN BLANCHETTE, SIMON CRUANES, AND UWE WALDMANN

Vrije Universiteit Amsterdam, Department of Computer Science, De Boelelaan 1111, 1081 HV Amsterdam, The Netherlands
e-mail address: a.bentkamp@vu.nl

Vrije Universiteit Amsterdam, Department of Computer Science, De Boelelaan 1111, 1081 HV Amsterdam, The Netherlands
e-mail address: j.c.blanchette@vu.nl

Aesthetic Integration, 600 Congress Ave., Austin, Texas, 78701, USA
e-mail address: simon@imandra.ai

Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
e-mail address: uwe@mpi-inf.mpg.de

Abstract. We introduce refutationally complete superposition calculi for intentional and extensional clausal λ-free higher-order logic, two formalisms that allow partial application and applied variables. The calculi are parameterized by a term order that need not be fully monotonic, making it possible to employ the λ-free higher-order lexicographic path and Knuth–Bendix orders. We implemented the calculi in the Zipperposition prover and evaluated them on Isabelle/HOL and TPTP benchmarks. They appear promising as a stepping stone towards complete, highly efficient automatic theorem provers for full higher-order logic.

1. Introduction

Superposition is a highly successful calculus for reasoning about first-order logic with equality. We are interested in *graceful* generalizations to higher-order logic: calculi that, as much as possible, coincide with standard superposition on first-order problems and that scale up to arbitrary higher-order problems.

As a stepping stone towards full higher-order logic, in this article we restrict our attention to a clausal λ-free fragment of polymorphic higher-order logic that supports partial application and application of variables (Section 2). This formalism is expressive enough to permit the axiomatization of higher-order combinators such as $\mathsf{pow} : \Pi\alpha.\ \mathsf{nat} \to (\alpha \to \alpha) \to \alpha \to \alpha$ (intended to denote the iterated application $h^n\ x$):

$$\mathsf{pow}\langle\alpha\rangle\ \mathsf{Zero}\ h \approx \mathsf{id}\langle\alpha\rangle \qquad\qquad \mathsf{pow}\langle\alpha\rangle\ (\mathsf{Succ}\ n)\ h\ x \approx h\ (\mathsf{pow}\langle\alpha\rangle\ n\ h\ x)$$

Key words and phrases: superposition calculus, clausal lambda-free higher-order logic, refutational completeness.

Extended version of Bentkamp et al., "Superposition for lambda-free higher-order logic" [11].

Preprint submitted to
Logical Methods in Computer Science

Conventionally, functions are applied without parentheses and commas, and variables are italicized. Notice the variable number of arguments to $\mathsf{pow}\langle\alpha\rangle$ and the application of h. The expressiveness of full higher-order logic can be recovered by introducing SK-style combinators to represent λ-abstractions and proxies for the logical symbols [42, 53].

A widespread technique to support partial application and application of variables in first-order logic is to make all symbols nullary and to represent application of functions by a distinguished binary symbol $\mathsf{app} : \Pi\alpha, \beta.\, \mathsf{fun}(\alpha, \beta) \times \alpha \to \beta$, where fun is an uninterpreted binary type constructor. Following this scheme, the higher-order term $\mathsf{f}(h\,\mathsf{f})$, where $\mathsf{f} : \kappa \to \kappa'$, is translated to $\mathsf{app}(\mathsf{f}, \mathsf{app}(h, \mathsf{f}))$—or rather $\mathsf{app}\langle\kappa, \kappa'\rangle(\mathsf{f}, \mathsf{app}\langle\mathsf{fun}(\kappa, \kappa'), \kappa\rangle(h, \mathsf{f}))$ if we specify the type arguments. We call this the *applicative encoding*. The existence of such a reduction to first-order logic explains why λ-free higher-order terms are also called "applicative first-order terms." Unlike for full higher-order logic, most general unifiers are unique for our λ-free fragment, just as they are for applicatively encoded first-order terms.

Although the applicative encoding is complete [42] and is employed fruitfully in tools such as HOLyHammer and Sledgehammer [18], it suffers from a number of weaknesses, all related to its gracelessness. Transforming all the function symbols into constants considerably restricts what can be achieved with term orders; for example, argument tuples cannot be compared using different methods for different symbols [43, Section 2.3.1]. In a prover, the encoding also clutters the data structures, slows down the algorithms, and neutralizes the heuristics that look at the terms' root symbols. But our chief objection is the sheer clumsiness of encodings and their poor integration with interpreted symbols. And they quickly accumulate; for example, using the traditional encoding of polymorphism relying on a distinguished binary function symbol t [17, Section 3.3] in conjunction with the applicative encoding, the term $\mathsf{Succ}\ x$ becomes $\mathsf{t}(\mathsf{nat}, \mathsf{app}(\mathsf{t}(\mathsf{fun}(\mathsf{nat}, \mathsf{nat}), \mathsf{Succ}), \mathsf{t}(\mathsf{nat}, x)))$. The term's simple structure is lost in translation.

Hybrid schemes have been proposed to strengthen the applicative encoding: If a given symbol always occurs with at least k arguments, these can be passed directly [47]. However, this relies on a closed-world assumption: that all terms that will ever be compared arise in the input problem. This noncompositionality conflicts with the need for complete higher-order calculi to synthesize arbitrary terms during proof search [12]. As a result, hybrid encodings are not an ideal basis for higher-order automated reasoning.

Instead, we propose to generalize the superposition calculus to *intensional* and *extensional* clausal λ-free higher-order logic. For the extensional version of the logic, the property $(\forall x.\, h\, x \approx k\, x) \longrightarrow h \approx k$ holds for all functions h, k of the same type. For each logic, we present two calculi (Section 3). The intentional calculi perfectly coincide with standard superposition on first-order clauses; the extensional calculi depend on an extra axiom.

Superposition is parameterized by a term order, which is used to prune the search space. If we assume that the term order is a simplification order enjoying totality on ground terms (i.e., terms containing no term or type variables), the standard calculus rules and completeness proof can be lifted verbatim. The only necessary changes concern the basic definitions of terms and substitutions. However, there is one monotonicity property that is hard to obtain unconditionally: *compatibility with arguments*. It states that $s' \succ s$ implies $s'\, t \succ s\, t$ for all terms s, s', t such that $s\, t$ and $s'\, t$ are well typed. Blanchette, Waldmann, and colleagues recently introduced graceful generalizations of the lexicographic path order (LPO) [20] and the Knuth–Bendix order (KBO) [6] with argument coefficients, but they both lack this property. For example, given a KBO with $\mathsf{g} \succ \mathsf{f}$, it may well be that $\mathsf{g}\ \mathsf{a} \prec \mathsf{f}\ \mathsf{a}$ if f has a large enough multiplier on its argument.

Our superposition calculi are designed to be refutationally complete for such nonmonotonic orders (Section 4). To achieve this, they include an inference rule for argument congruence, which derives $C \vee s\ x \approx t\ x$ from $C \vee s \approx t$. The redundancy criterion is defined in such a way that the larger, derived clause is not subsumed by the premise. In the completeness proof, the most difficult case is the one that normally excludes superposition at or below variables using the induction hypothesis. With nonmonotonicity, this approach no longer works, and we propose two alternatives: Either perform some superposition inferences into higher-order variables or "purify" the clauses to circumvent the issue. We refer to the corresponding calculi as *nonpurifying* and *purifying*.

The calculi are implemented in the Zipperposition prover [29] (Section 5). We evaluate them on first- and higher-order Isabelle/HOL [23] and TPTP benchmarks [62, 63] and compare them with the applicative encoding (Section 6). We find that there is a substantial cost associated with the applicative encoding, that the nonmonotonicity is not particularly expensive, and that the nonpurifying calculi outperform the purifying variants.

An earlier version of this work was presented at IJCAR 2018 [11]. This article extends the conference paper with detailed soundness and completeness proofs and more explanations. Because of too weak selection restrictions on the purifying variants, our claim of refutational completeness in the conference version was not entirely correct. We now strengthened the selection restrictions accordingly. Moreover, we extended the logic with polymorphism, leading to minor modifications to the calculus. We also simplified the presentation of the clausal fragment of the logic that interests us. In particular, we removed mandatory arguments. The redundancy criterion also differs slightly from the conference version. Finally, we updated the empirical evaluation to reflect recent improvements in the Zipperposition prover.

2. Logic

Our logic is intended as a convenient intermediate step on the way towards full higher-order logic (also called simple type theory) [27, 36]. Refutational completeness of calculi for higher-order logic is usually stated in terms of Henkin semantics [12, 38], in which the universes used to interpret functions need only contain the functions that can be expressed as terms. Since the terms of λ-free higher-order logic exclude λ-abstractions, in "λ-free Henkin semantics" the universes interpreting functions can be even smaller. In that sense, our semantics resemble Henkin prestructures [45, Section 5.4]. In contrast to other higher-order logics [64], there are no comprehension principles, and we disallow nesting of Boolean formulas inside terms.

2.1. Syntax. We fix a set Σ_{ty} of type constructors with arities and a set $\mathcal{V}_{\mathsf{ty}}$ of type variables. We require at least one nullary type constructor and a binary type constructor $\to$ to be present in Σ_{ty}. Types τ, υ of λ-free higher-order logic are either a type variable $\alpha \in \mathcal{V}_{\mathsf{ty}}$ or of the form $\kappa(\bar{\tau}_n)$ for an n-ary type constructor $\kappa \in \Sigma_{\mathsf{ty}}$ and types $\bar{\tau}_n$. Here and elsewhere, we write $\bar{a}_n$ or $\bar{a}$ to abbreviate the tuple $(a_1, \ldots, a_n)$ or product $a_1 \times \cdots \times a_n$, for $n \geq 0$. We write κ for $\kappa()$ and $\tau \to \upsilon$ for $\to(\tau, \upsilon)$. A type declaration is an expression of the form $\Pi\bar{\alpha}_m.\ \tau$ (or simply τ if if $m = 0$), where all type variables occurring in τ belong to $\bar{\alpha}_m$.

We fix a set Σ of symbols with type declarations, written as $\mathsf{f} : \Pi\bar{\alpha}_m.\ \tau$ or f, and a set $\mathcal{V}$ of typed variables, written as $x : \tau$ or x. To avoid empty Herbrand universes, we require Σ to contain a symbol with type declaration $\Pi\alpha.\ \alpha$. The sets $(\Sigma_{\mathsf{ty}}, \mathcal{V}_{\mathsf{ty}}, \Sigma, \mathcal{V})$ form the logic's signature. We reserve the letters s, t, u, v, w for terms and x, y, z for variables and write $: \tau$ to indicate their type. The set of λ-free higher-order terms is defined inductively as follows.

Every variable in X is a term. If $\mathsf{f} : \Pi\bar{\alpha}_m.\ \tau$ is a symbol and $\bar{\upsilon}_m$ are types, then $\mathsf{f}\langle\bar{\upsilon}_m\rangle : \tau$ is a term. If $t : \tau \to \upsilon$ and $u : \tau$, then $t\ u : \upsilon$ is a term, called an *application*. Non-application terms are called *heads*. A term is *ground* if it is built without using type or term variables. Using the spine notation [26], terms can be decomposed in a unique way as a head t applied to zero or more arguments: $t\, s_1 \ldots s_n$ or $t\, \bar{s}_n$ (abusing notation). Substitution and unification are generalized in the obvious way, without the complexities associated with λ-abstractions; for example, the most general unifier of $x\ \mathsf{b}\ z$ and $\mathsf{f}\ \mathsf{a}\ y\ \mathsf{c}$ is $\{x \mapsto \mathsf{f}\ \mathsf{a}, y \mapsto \mathsf{b}, z \mapsto \mathsf{c}\}$, and that of $h\ (\mathsf{f}\ \mathsf{a})$ and $\mathsf{f}\ (h\ \mathsf{a})$ is $\{h \mapsto \mathsf{f}\}$.

An equation $s \approx t$ is formally an unordered pair of terms s and t. A literal is an equation or a negated equation, written $\neg\ s \approx t$ or $s \not\approx t$. A clause $L_1 \vee \cdots \vee L_n$ is a finite multiset of literals L_j. The empty clause is written as $\bot$.

2.2. Semantics. A *type interpretation* $\mathcal{I}_{\mathsf{ty}} = (\mathcal{U}, \mathcal{J}_{\mathsf{ty}})$ is defined as follows. The set $\mathcal{U}$ is a nonempty collection of nonempty sets, called *universes*. The function $\mathcal{J}_{\mathsf{ty}}$ associates a function $\mathcal{J}_{\mathsf{ty}}(\kappa) : \mathcal{U}^n \to \mathcal{U}$ with each n-ary type constructor κ. A *type valuation* ξ is a function that maps every type variable to a universe. The *denotation* of a type for a type interpretation $\mathcal{I}_{\mathsf{ty}}$ and a type valuation ξ is defined by $[\![\alpha]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} = \xi(\alpha)$ and $[\![\kappa(\bar{\tau})]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} = \mathcal{J}_{\mathsf{ty}}(\kappa)([\![\bar{\tau}]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}})$. Here and elsewhere, we abuse notation by applying an operation on a tuple when it must be applied elementwise; thus, $[\![\bar{\tau}_n]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$ stands for $[\![\tau_1]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}, \ldots, [\![\tau_n]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$.

A type valuation ξ can be extended to be a *valuation* by additionally assigning an element $\xi(x) \in [\![\tau]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$ to each variable $x : \tau$. An *interpretation function* $\mathcal{J}$ for a type interpretation $\mathcal{I}_{\mathsf{ty}}$ associates with each symbol $\mathsf{f} : \Pi\bar{\alpha}_m.\ \tau$ and universe tuple $\bar{U}_m \in \mathcal{U}^m$ a value $\mathcal{J}(\mathsf{f}, \bar{U}_m) \in [\![\tau]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$, where ξ is the type valuation that maps each α_i to U_i. Loosely following Fitting [35, Section 2.5], an *extension function* $\mathcal{E}$ associates to any pair of universes $U_1, U_2 \in \mathcal{U}$ a function $\mathcal{E}_{U_1,U_2} : \mathcal{J}_{\mathsf{ty}}(\to)(U_1, U_2) \to (U_1 \to U_2)$. Together, a type interpretation, an interpretation function, and an extension function form an *interpretation* $\mathcal{I} = (\mathcal{U}, \mathcal{J}_{\mathsf{ty}}, \mathcal{J}, \mathcal{E})$.

An interpretation is *extensional* if $\mathcal{E}_{U_1,U_2}$ is injective for all U_1, U_2. Both intensional and extensional logics are widely used for interactive theorem proving; for example, Coq's calculus of inductive constructions is intensional [14], whereas Isabelle/HOL is extensional [49]. The semantics is *standard* if $\mathcal{E}_{U_1,U_2}$ is bijective for all U_1, U_2.

For an interpretation $\mathcal{I} = (\mathcal{U}, \mathcal{J}_{\mathsf{ty}}, \mathcal{J}, \mathcal{E})$ and a valuation ξ, the denotation of a term is defined as follows: For variables x, let $[\![x]\!]^{\xi}_{\mathcal{I}} = \xi(x)$. For symbols f, let $[\![\mathsf{f}\langle\bar{\tau}\rangle]\!]^{\xi}_{\mathcal{I}} = \mathcal{J}(\mathsf{f}, [\![\bar{\tau}]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}})$. For applications $s\ t$ of a term $s : \tau \to \upsilon$ to a term $t : \tau$, let $U_1 = [\![\bar{\tau}]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$, $U_2 = [\![\bar{\upsilon}]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$, and $[\![s\ t]\!]^{\xi}_{\mathcal{I}} = \mathcal{E}_{U_1,U_2}([\![s]\!]^{\xi}_{\mathcal{I}})([\![t]\!]^{\xi}_{\mathcal{I}})$. If t is a ground term, we also write $[\![t]\!]_{\mathcal{I}}$ for the denotation of t because it does not depend on the valuation.

An equation $s \approx t$ is true in $\mathcal{I}$ for ξ if $[\![s]\!]^{\xi}_{\mathcal{I}} = [\![t]\!]^{\xi}_{\mathcal{I}}$; otherwise, it is false. A disequation $s \not\approx t$ is true if $s \approx t$ is false. A clause is true if at least one of its literals is true. The interpretation $\mathcal{I}$ is a model of a clause C, written $\mathcal{I} \models C$, if C is true in $\mathcal{I}$ for all valuations ξ. It is a model of a set of clauses if it is a model of all contained clauses.

For example, given the signature $(\Sigma_{\mathsf{ty}}, \mathcal{V}_{\mathsf{ty}}, \Sigma, \mathcal{V}) = (\{\kappa, \to\}, \{\}, \{\mathsf{a} : \kappa\}, \{h : \kappa \to \kappa\})$, the clause $h\ \mathsf{a} \not\approx \mathsf{a}$ has an extensional model with $\mathcal{U} = \{U_1, U_2\}$, $U_1 = \{a, b\}$ $(a \neq b)$, $U_2 = \{f\}$, $\mathcal{J}_{\mathsf{ty}}(\kappa) = U_1$, $\mathcal{J}_{\mathsf{ty}}(\to)(U_1, U_1) = U_2$, $\mathcal{J}(\mathsf{a}) = a$, $\mathcal{E}_{U_1,U_1}(f)(a) = \mathcal{E}_{U_1,U_1}(f)(b) = b$.

3. The Inference Systems

We introduce four versions of the *clausal λ-free higher-order superposition calculus*, varying along two axes: intentional versus extensional, and nonpurifying versus purifying. To avoid repetitions, our presentation unifies them into a single framework.

3.1. The Inference Rules. The calculi are parameterized by a partial order $\succ$ on ground terms that is well founded and total and that has the subterm property. It must also be *compatible with green contexts*, meaning that $t' \succ t$ implies $s\ t'\ \bar{u} \succ s\ t\ \bar{u}$. On the other hand, it need not be *compatible with arguments*: $s' \succ s$ need not imply $s'\ t \succ s\ t$. Green contexts are built around *green subterms*, defined inductively as follows. A term t' is a green subterm of t if either $t = t'$; or $t = s\ \bar{u}$ and t' is a green subterm of u_i for some i. We write $s\langle u\rangle$ to indicate that the subterm u of $s[u]$ is a green subterm; correspondingly, $s[\,]$ is a green context. For example, f and $\mathsf{f}\ \mathsf{a}$ are subterms of $\mathsf{f}\ \mathsf{a}\ \mathsf{b}$, but not green subterms; correspondingly, $[\,]\ \mathsf{a}\ \mathsf{b}$ and $[\,]\ \mathsf{b}$ are not green contexts.

For nonground terms, the only requirement on $\succ$ is stability under grounding substitutions (i.e., $t \succ s$ implies $t\theta \succ s\theta$ for all substitutions θ grounding t and s). The literal and clause orders are defined from $\succ$ as multiset extensions in the standard way [4]. Despite their names, the term, literal, and clause orders need not be transitive on nonground entities.

Literal selection is supported. The selection function maps each clause C to a subclause of C consisting of negative literals. A literal L is (*strictly*) *eligible* w.r.t. a substitution σ in C if it is selected in C or there are no selected literals in C and $L\sigma$ is (strictly) maximal in $C\sigma$. If σ is the identity substitution, we leave it implicit.

The following four rules are common to all four calculi. We regard positive and negative superposition as two cases of the same rule

$$\frac{\overbrace{D' \vee t \approx t'}^{D} \qquad \overbrace{C' \vee [\neg]\, s\langle u\rangle \approx s'}^{C}}{(D' \vee C' \vee [\neg]\, s\langle t'\rangle \approx s')\sigma}\ \textsc{Sup}$$

where $\sigma = \mathrm{mgu}(t, u)$; $t\sigma \not\preceq t'\sigma$; $s\langle u\rangle\sigma \not\preceq s'\sigma$; $t \approx t'$ is strictly eligible w.r.t. σ in D; $s\langle u\rangle \approx s'$ is eligible w.r.t. σ in C and, if positive, even strictly eligible; and $C\sigma \not\preceq D\sigma$. Moreover, the *variable condition* must hold; it varies from one calculus to another and is specified below.

The equality resolution and equality factoring rules are almost identical to their standard counterparts:

$$\frac{\overbrace{C' \vee s \not\approx s'}^{C}}{C'\sigma}\ \textsc{EqRes} \qquad\qquad \frac{\overbrace{C' \vee s' \approx t' \vee s \approx t}^{C}}{(C' \vee t \not\approx t' \vee s \approx t')\sigma}\ \textsc{EqFact}$$

The side conditions for EqRes are $\sigma = \mathrm{mgu}(s, s')$ and $s \not\approx s'$ is eligible w.r.t. σ in C. The side conditions for EqFact are $\sigma = \mathrm{mgu}(s, s')$, $s'\sigma \not\preceq t'\sigma$, $s\sigma \not\preceq t\sigma$, and $s \approx t$ is eligible w.r.t. σ in C.

The following *argument congruence* rule compensates for the limitation that the superposition rule applies only to green subterms:

$$\frac{\overbrace{C' \vee s \approx s'}^{C}}{C'\sigma \vee (s\sigma)\ \bar{x} \approx (s'\sigma)\ \bar{x}}\ \textsc{ArgCong}$$

The literal $s \approx s'$ must be strictly eligible w.r.t. σ in C, and $\bar{x}$ is a nonempty tuple of distinct fresh variables. The substitution σ is the most general type substitution that ensures well-typedness of the conclusion. In particular, if s takes m arguments, there are m ArgCong conclusions for this literal, for which σ is the identity and $\bar{x}$ is a tuple of $1, \ldots, m-1$, or m variables. If the result type of s is a type variable, we have in addition infinitely many ArgCong conclusions, for which σ instantiates the type variable in the result type of s with with $\bar{\alpha}_k \to \beta$ for some $k > 0$ and fresh type variables $\bar{\alpha}_k$ and β and for which $\bar{x}$ is a tuple of $m + k$ variables.

For the **intensional nonpurifying** variant, the variable condition of the Sup rule is as follows: "Either $u \notin \mathcal{V}$ or there exists a grounding substitution θ with $t\sigma\theta \succ t'\sigma\theta$ and $C\sigma\theta \prec C\{u \mapsto t'\}\sigma\theta$." This condition generalizes the standard condition that $u \notin \mathcal{V}$. The two coincide if C is first-order or if the term order is monotonic. In some cases involving nonmonotonicity, the variable condition effectively mandates Sup inferences at variable positions of the right premise, but never below. We will call theses inferences *at variables*.

For the **extensional nonpurifying** calculus, the variable condition uses the following definition.

Definition 3.1. A term of the form $x\ \bar{s}_n$, for $n \geq 0$, *jells* with a literal $t \approx t' \in D$ if $t = \tilde{t}\ \bar{y}_n$ and $t' = \tilde{t}'\ \bar{y}_n$ for some terms $\tilde{t}$, $\tilde{t}'$ and distinct variables $\bar{y}_n$ that do not occur elsewhere in D.

Using the naming convention from Definition 3.1 for $\tilde{t}'$, the variable condition can be stated as follows: "If u has a variable head x and jells with the literal $t \approx t' \in D$, there must exist a grounding substitution θ with $t\sigma\theta \succ t'\sigma\theta$ and $C\sigma\theta \prec C''\sigma\theta$, where $C'' = C\{x \mapsto \tilde{t}'\}$." If C is first-order, this amounts to $u \notin \mathcal{V}$. Since the order is compatible with green contexts, the substitution θ can exist only if x occurs applied in C.

Moreover, the extensional nonpurifying calculus has one additional rule, the positive extensionality rule, and one axiom, the extensionality axiom. The rule is

$$\frac{C' \vee s\ \bar{x} \approx s'\ \bar{x}}{C' \vee s \approx s'}\ \textsc{PosExt}$$

where $\bar{x}$ is a tuple of distinct variables that do not occur in C', s, or s', and $s\ \bar{x} \approx s'\ \bar{x}$ is strictly eligible in the premise. The extensionality axiom uses a polymorphic Skolem symbol $\mathsf{diff} : \Pi\alpha, \beta.\ (\alpha \to \beta)^2 \to \alpha$ characterized by the axiom

$$x\ (\mathsf{diff}\langle\alpha, \beta\rangle\ x\ y) \not\approx y\ (\mathsf{diff}\langle\alpha, \beta\rangle\ x\ y) \vee x \approx y \qquad \text{(Ext)}$$

Unlike the nonpurifying calculi, the purifying calculi never perform superposition at variables. Instead, they rely on purification [24, 31, 54, 58] (also called abstraction) to circumvent nonmonotonicity. The idea is to rename apart problematic occurrences of a variable x in a clause to $x_1, \ldots, x_n$ and to add *purification literals* $x_1 \not\approx x$, $\ldots$, $x_n \not\approx x$ to connect the new variables to x. We must then ensure that all clauses are purified, by processing the initial clause set and the conclusion of every inference or simplification.

In the **intensional purifying** calculus, the purification $\mathit{pure}(C)$ of clause C is defined as the result of the following procedure. Choose a variable x that occurs applied in C and also unapplied in a literal of C that is not of the form $x \not\approx y$. If no such variable exists, terminate. Otherwise, replace all unapplied occurrences of x in C by a fresh variable x' and add the purification literal $x' \not\approx x$. Then repeat the procedure with another variable. For example,

$$\mathit{pure}(x\ \mathsf{a} \approx x\ \mathsf{b} \vee \mathsf{f}\ x \approx \mathsf{g}\ x) = x\ \mathsf{a} \approx x\ \mathsf{b} \vee \mathsf{f}\ x' \approx \mathsf{g}\ x' \vee x \not\approx x'$$

The variable condition is "$u \notin \mathcal{V}$." The conclusion C of ArgCong is changed to $\mathit{pure}(C)$; the other rules preserve purity of their premises.

In the **extensional purifying** calculus, $\mathit{pure}(C)$ is defined as follows. Choose a variable x occurring in green subterms $x\ \bar{u}$ and $x\ \bar{v}$ in literals of C that are not of the form $x \not\approx y$, where $\bar{u}$ and $\bar{v}$ are distinct (possibly empty) term tuples. If no such variable exists, terminate. Otherwise, replace all green subterms $x\ \bar{v}$ with $x'\,\bar{v}$, where x' is fresh, and add the purification literal $x' \not\approx x$. Then repeat the procedure until no variable fulfilling the requirements is left. For example,

$$\mathit{pure}(x\ \mathsf{a} \approx x\ \mathsf{b} \vee \mathsf{f}\ x \approx \mathsf{g}\ x) = x\ \mathsf{a} \approx x'\ \mathsf{b} \vee \mathsf{f}\ x'' \approx \mathsf{g}\ x'' \vee x' \not\approx x \vee x'' \not\approx x$$

Like the extensional nonpurifying calculus, this calculus variant also contains the PosExt rule and axiom (Ext) introduced above. The variable condition is "either u has a non-variable head or u does not jell with the literal $t \approx t' \in D$." The conclusion E of each rule is changed to $\mathit{pure}(E)$, except for PosExt, which preserves purity.

Finally, we impose further restrictions on literal selection. In the nonpurifying variants, a literal may not be selected if $x\ \bar{u}$ is a maximal term of the clause and the literal contains a green subterm $x\ \bar{v}$ with $\bar{v} \neq \bar{u}$. In the purifying calculi, a literal may not be selected if it contains a variable of functional type. These restrictions are needed for our completeness proof. It might be possible to avoid them at the cost of a more elaborate argument.

Remark 3.2. In descriptions of first-order logic with equality, the property $y \approx y' \longrightarrow \mathsf{f}(\bar{x}, y, \bar{z}) \approx \mathsf{f}(\bar{x}, y', \bar{z})$ is often referred to as "function congruence." It seems natural to use the same label for the higher-order version $t \approx t' \longrightarrow s\ t \approx s\ t'$ and to call the companion property $s \approx s' \longrightarrow s\ t \approx s'\ t$ "argument congruence," whence the name ArgCong for our inference rule. This nomenclature is far from universal; for example, the Isabelle/HOL theorem *fun_cong* captures argument congruence and *arg_cong* captures function congruence.

3.2. Rationale for the Inference Rules. A key restriction of all four calculi is that they superpose only at green subterms, mirroring the term order's compatibility with green contexts. The ArgCong rule then makes it possible to simulate superposition at non-green subterms. However, in conjunction with the Sup rules, ArgCong can exhibit an unpleasant behavior, which we call *argument congruence explosion*:

$$\textsc{ArgCong}\ \dfrac{\textsc{Sup}\ \dfrac{\mathsf{g} \approx \mathsf{f}}{\mathsf{g}\ x \approx \mathsf{f}\ x \quad h\ \mathsf{a} \not\approx \mathsf{b}}}{\mathsf{f}\ \mathsf{a} \not\approx \mathsf{b}} \qquad \textsc{ArgCong}\ \dfrac{\textsc{Sup}\ \dfrac{\mathsf{g} \approx \mathsf{f}}{\mathsf{g}\ x\ y\ z \approx \mathsf{f}\ x\ y\ z \quad h\ \mathsf{a} \not\approx \mathsf{b}}}{\mathsf{f}\ x\ y\ \mathsf{a} \not\approx \mathsf{b}}$$

In both derivation trees, the higher-order variable h is effectively the target of a Sup inference. Such derivations essentially amount to superposition at variable positions (as shown on the left) or even superposition below variable positions (as shown on the right), both of which can be extremely prolific. In standard superposition, the explosion is averted by the condition on the Sup rule that $u \notin \mathcal{V}$. In the extensional purifying calculus, the variable condition tests that either u has a non-variable head or u does not jell with the literal $t \approx t' \in D$, which prevents derivations such as the above. In the corresponding nonpurifying variant, some such derivations may need to be performed when the term order exhibits nonmonotonicity for the terms of interest.

In the intensional calculi, the explosion can arise because the variable conditions are weaker. The following example shows that the intensional nonpurifying calculus would be incomplete if we used the variable condition of the extensional nonpurifying calculus.

Example 3.3. Consider a left-to-right LPO [20] instance with precedence $\mathsf{h} \succ \mathsf{g} \succ \mathsf{f} \succ \mathsf{b} \succ \mathsf{a}$, and consider the following unsatisfiable clause set:

$$\mathsf{h}\ x \approx \mathsf{f}\ x \qquad \mathsf{g}\ (x\ \mathsf{b})\ x \approx \mathsf{a} \qquad \mathsf{g}\ (\mathsf{f}\ \mathsf{b})\ \mathsf{h} \not\approx \mathsf{a}$$

The only possible inference is a Sup inference of the first into the second clause, but the variable condition of the extensional nonpurifying calculus is not met.

It is unclear whether the variable condition of the intensional purifying calculus could be strengthened, but our completeness proof suggests that it cannot.

The variable conditions in the extensional calculi are designed to prevent the argument congruence explosion shown above, but since they consider only the shape of the clauses, they might also block Sup inferences whose side premises do not originate from ArgCong. This is why we need the PosExt rule.

Example 3.4. In the following unsatisfiable clause set, the only possible inference from these clauses in the extensional nonpurifying calculus is PosExt, showing its necessity:

$$\mathsf{g}\ x \approx \mathsf{f}\ x \qquad \mathsf{g} \not\approx \mathsf{f} \qquad x\ (\mathsf{diff}\langle \alpha, \beta\rangle\ x\ y) \not\approx y\ (\mathsf{diff}\langle \alpha, \beta\rangle\ x\ y) \vee x \approx y$$

The same argument applies for the purifying variant with the difference that the third clause must be purified.

Due to nonmonotonicity, for refutational completeness we need either to purify the clauses or to allow some superposition at variable positions, as mandated by the respective variable conditions. Without either of these measures, at least the extensional calculi and presumably also the intensional calculi would be incomplete, as the next example demonstrates.

Example 3.5. Consider the following clause set:

$$\mathsf{k}\ (\mathsf{g}\ x) \approx \mathsf{k}\ (x\ \mathsf{b}) \quad \mathsf{k}\ (\mathsf{f}\ (\mathsf{h}\ \mathsf{a})\ \mathsf{b}) \not\approx \mathsf{k}\ (\mathsf{g}\ \mathsf{h}) \quad \mathsf{f}\ (\mathsf{h}\ \mathsf{a}) \approx \mathsf{h} \quad \mathsf{f}\ (\mathsf{h}\ \mathsf{a})\ x \approx \mathsf{h}\ x$$
$$x\ (\mathsf{diff}\langle \alpha, \beta\rangle\ x\ y) \not\approx y\ (\mathsf{diff}\langle \alpha, \beta\rangle\ x\ y) \vee x \approx y$$

Using a left-to-right LPO [20] instance with precedence $\mathsf{k} \succ \mathsf{h} \succ \mathsf{g} \succ \mathsf{f} \succ \mathsf{b} \succ \mathsf{a}$, this clause set is saturated w.r.t. the extensional purifying calculus when omitting purification. It also quickly saturates using the extensional nonpurifying calculus when omitting Sup inferences at variables. By contrast, the intensional variants derive $\bot$, even without purification and without Sup inferences at variables, because of the less restrictive variable conditions.

This raises the question as to whether the intensional variants actually need to purify or to perform Sup inferences at variables. Omitting purification and Sup inferences at variables in the intensional calculi is complete when redundant clauses are kept, but we conjecture that it is incomplete in general.

We initially considered inference rules instead of axiom (Ext). However, we did not find a set of inference rules that is complete and leads to fewer inferences than (Ext). We considered the PosExt rule described above in combination with the following rule:

$$\frac{C \vee s \not\approx t}{C \vee s\ (\mathsf{sk}\langle \bar{\alpha}\rangle\ \bar{x}_n) \not\approx t\ (\mathsf{sk}\langle \bar{\alpha}\rangle\ \bar{x}_n)}\ \textsc{NegExt}$$

where sk is a fresh Skolem symbol and $\bar{\alpha}$ and $\bar{x}_n$ are the type and term variables occurring free in the the literal $s \not\approx t$. However, these two rules do not suffice for a refutationally complete calculus, as the following example demonstrates:

Example 3.6. Consider the clause set

$$\mathsf{f}\ x \approx \mathsf{a} \qquad \mathsf{g}\ x \approx \mathsf{a} \qquad \mathsf{h}\ \mathsf{f} \approx \mathsf{b} \qquad \mathsf{h}\ \mathsf{g} \not\approx \mathsf{b}$$

Assuming that all four equations are oriented from left to right, this set is saturated w.r.t. the extensional calculi if (Ext) is replaced by NegExt; yet it is unsatisfiable in an extensional logic.

Example 3.7. A significant advantage of our calculi over the use of standard superposition on applicatively encoded problems is the flexibility they offer in orienting equations. The following equations provide two definitions of addition on Peano numbers:

$$\mathsf{add_L}\ \mathsf{Zero}\ y \approx y \qquad \mathsf{add_R}\ x\ \mathsf{Zero} \approx x$$
$$\mathsf{add_L}\ (\mathsf{Succ}\ x)\ y \approx \mathsf{add_L}\ x\ (\mathsf{Succ}\ y) \qquad \mathsf{add_R}\ x\ (\mathsf{Succ}\ y) \approx \mathsf{add_R}\ (\mathsf{Succ}\ x)\ y$$

Let $\mathsf{add_L}\ (\mathsf{Succ}^{100}\ \mathsf{Zero})\ \mathsf{n} \not\approx \mathsf{add_R}\ \mathsf{n}\ (\mathsf{Succ}^{100}\ \mathsf{Zero})$ be the negated conjecture. With LPO, we can use a left-to-right comparison for $\mathsf{add_L}$'s arguments and a right-to-left comparison for $\mathsf{add_R}$'s arguments to orient all four equations from left to right. Then the negated conjecture can be simplified to $\mathsf{Succ}^{100}\ \mathsf{n} \not\approx \mathsf{Succ}^{100}\ \mathsf{n}$ by simplification (demodulation), and $\bot$ can be derived with a single inference. If we use the applicative encoding instead, there is no instance of LPO or KBO that can orient both recursive equations from left to right. For at least one of the two sides of the negated conjecture, simplification is replaced by 100 Sup inferences, which is much less efficient, especially in the presence of additional axioms.

3.3. Soundness. To show the inferences' soundness, we need the substitution lemma for our logic:

Lemma 3.8 (Substitution lemma)**.** *Let $\mathcal{I} = (\mathcal{U}, \mathcal{J}_{\mathsf{ty}}, \mathcal{J}, \mathcal{E})$ be a λ-free higher-order interpretation. Then*

$$[\![\tau\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} = [\![\tau]\!]^{\xi'}_{\mathcal{I}_{\mathsf{ty}}} \text{ and } [\![t\rho]\!]^{\xi}_{\mathcal{I}} = [\![t]\!]^{\xi'}_{\mathcal{I}}$$

for all terms t, all types τ, and all substitutions ρ, where $\xi'(\alpha) = [\![\alpha\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$ for all type variables α and $\xi'(x) = [\![x\rho]\!]^{\xi}_{\mathcal{I}}$ for all term variables x.

Proof. First, we prove that $[\![\tau\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} = [\![\tau]\!]^{\xi'}_{\mathcal{I}_{\mathsf{ty}}}$ by induction on the structure of τ. If $\tau = \alpha$ is a type variable,

$$[\![\alpha\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} = \xi'(\alpha) = [\![\alpha]\!]^{\xi'}_{\mathcal{I}_{\mathsf{ty}}}$$

If $\tau = \kappa(\bar{\upsilon})$ for some type constructor κ and types $\bar{\upsilon}$,

$$[\![\kappa(\bar{\upsilon})\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} = \mathcal{J}_{\mathsf{ty}}(\kappa)([\![\bar{\upsilon}\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}) \overset{\text{IH}}{=} \mathcal{J}_{\mathsf{ty}}(\kappa)([\![\bar{\upsilon}]\!]^{\xi'}_{\mathcal{I}_{\mathsf{ty}}}) = [\![\kappa(\bar{\upsilon})]\!]^{\xi'}_{\mathcal{I}_{\mathsf{ty}}}$$

Next, we prove $[\![t\rho]\!]^{\xi}_{\mathcal{I}} = [\![t]\!]^{\xi'}_{\mathcal{I}}$ by structural induction on t. If $t = y$, then by the definition of the denotation of a variable

$$[\![y\rho]\!]^{\xi}_{\mathcal{I}} = \xi'(y) = [\![y]\!]^{\xi'}_{\mathcal{I}}$$

If $t = \mathsf{f}\langle\bar{\tau}\rangle$, then by the definition of the term denotation

$$[\![\mathsf{f}\langle\bar{\tau}\rangle\rho]\!]^{\xi}_{\mathcal{I}} = \mathcal{J}(\mathsf{f}, [\![\bar{\tau}\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}) \overset{\text{IH}}{=} \mathcal{J}(\mathsf{f}, [\![\bar{\tau}]\!]^{\xi'}_{\mathcal{I}_{\mathsf{ty}}}) = [\![\mathsf{f}\langle\bar{\tau}\rangle]\!]^{\xi'}_{\mathcal{I}}$$

If $t = u\ v$, then by the definition of the term denotation

$$[\![(u\ v)\rho]\!]^{\xi}_{\mathcal{I}} = \mathcal{E}_{U_1,U_2}([\![u\rho]\!]^{\xi}_{\mathcal{I}})([\![v\rho]\!]^{\xi}_{\mathcal{I}}) \overset{\text{IH}}{=} \mathcal{E}_{U_1,U_2}([\![u]\!]^{\xi'}_{\mathcal{I}})([\![v]\!]^{\xi'}_{\mathcal{I}}) = [\![u\ v]\!]^{\xi'}_{\mathcal{I}}$$

where u is of type $\tau \to \upsilon$, $U_1 = [\![\tau\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} \overset{\text{IH}}{=} [\![\tau]\!]^{\xi'}_{\mathcal{I}_{\mathsf{ty}}}$, and $U_2 = [\![\upsilon\rho]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} \overset{\text{IH}}{=} [\![\upsilon]\!]^{\xi'}_{\mathcal{I}_{\mathsf{ty}}}$. □

Lemma 3.9. *If $\mathcal{I} \models C$ for some interpretation $\mathcal{I}$ and some clause C, then $\mathcal{I} \models C\rho$ for all substitutions ρ.*

Proof. We need to show that C is true in $\mathcal{I}$ for all valuations ξ. Given a valuation ξ, define ξ' as in Lemma 3.8. Then, by Lemma 3.8, a literal in $C\rho$ is true in $\mathcal{I}$ for ξ if and only if the corresponding literal in C is true in $\mathcal{I}$ for ξ'. There must be at least one such literal because $\mathcal{I} \models C$ and hence C is in particular true in $\mathcal{I}$ for ξ'. Therefore, $C\rho$ is true in $\mathcal{I}$ for ξ. □

Theorem 3.10 (Soundness of the intensional calculi)**.** *The inference rules* Sup, EqRes, EqFact, *and* ArgCong *are sound* (*even without the variable condition and the side conditions on order and eligibility*).

Proof. We fix an inference and an interpretation $\mathcal{I}$ that is a model of the premises. We need to show that it is also a model of the conclusion.

From the definition of the denotation of a term, it is obvious that congruence holds at all subterms in our logic. By Lemma 3.9, $\mathcal{I}$ is a model of the σ-instances of the premises as well, where σ is the substitution used for the inference. Fix a valuation ξ. By making case distinctions on the truth in $\mathcal{I}$ under ξ of the literals of the σ-instances of the premises, using the conditions that σ is a unifier, and applying congruence, it follows that the conclusion is also true in $\mathcal{I}$ under ξ. □

Theorem 3.11 (Soundness of the extensional calculi)**.** *The inference rules* Sup, EqRes, EqFact, ArgCong, *and* PosExt *are sound w.r.t. extensional interpretations* (*even without the variable condition and the side conditions on order and eligibility*).

Proof. We only need to prove soundness of PosExt. For the other rules, we can proceed as in Theorem 3.10. By induction on the length of $\bar{x}$, it suffices to prove soundness of PosExt for one variable x instead of a tuple $\bar{x}$. We fix an inference and an extensional interpretation $\mathcal{I}$ that is a model of the premise $C' \vee s\, x \approx s'\, x$. We need to show that it is also a model of the conclusion $C' \vee s \approx s'$.

Let ξ be a valuation. If C' is true in $\mathcal{I}$ under ξ, the conclusion is clearly true as well. Otherwise C' is false in $\mathcal{I}$ under ξ, and also under $\xi[x \mapsto a]$ for all a because x does not occur in C'. Since the premise is true in $\mathcal{I}$, $s\, x = s'\, x$ must be true in $\mathcal{I}$ under $\xi[x \mapsto a]$ for all a. Hence, for appropriate universes U_1, U_2, we have $\mathcal{E}_{U_1,U_2}([\![s]\!]^{\xi[x\mapsto a]}_{\mathcal{I}})(a) = [\![s\, x]\!]^{\xi[x\mapsto a]}_{\mathcal{I}} = [\![s'\, x]\!]^{\xi[x\mapsto a]}_{\mathcal{I}} = \mathcal{E}_{U_1,U_2}([\![s']\!]^{\xi[x\mapsto a]}_{\mathcal{I}})(a)$. Since s and s' do not contain x, $[\![s]\!]^{\xi[x\mapsto a]}_{\mathcal{I}}$ and $[\![s']\!]^{\xi[x\mapsto a]}_{\mathcal{I}}$ do not depend on a. Thus, $\mathcal{E}_{U_1,U_2}([\![s]\!]^{\xi}_{\mathcal{I}}) = \mathcal{E}_{U_1,U_2}([\![s']\!]^{\xi}_{\mathcal{I}})$. Since $\mathcal{I}$ is extensional, $\mathcal{E}_{U_1,U_2}$ is injective and hence $[\![s]\!]^{\xi}_{\mathcal{I}} = [\![s']\!]^{\xi}_{\mathcal{I}}$. It follows that $s \approx s'$ is true in $\mathcal{I}$ under ξ, and so is the entire conclusion of the inference. □

A problem expressed in higher-order logic must be transformed into clausal normal form before the calculi can be applied. This process works as in the first-order case, except for skolemization. The issue is that skolemization, when performed naively, is unsound for higher-order logic with a Henkin semantics [48, Section 6], because it introduces new functions that can be used to instantiate variables.

The core of this article is not affected by this because the problems are given in clausal form. For the implementation, we claim soundness only w.r.t. models that satisfy the axiom of choice, which is the semantics mandated by the TPTP THF format [62]. By contrast, refutational completeness holds w.r.t. arbitrary models as defined above. Alternatively,

skolemization can be made sound by introducing mandatory arguments as described by Miller [48, Section 6] and in the conference version of this article [11].

This issue also affects axiom (Ext) because it contains the Skolem symbol diff. As a consequence, (Ext) does not hold in all extensional interpretations. The extensional calculi are thus only sound w.r.t. interpretations in which (Ext) holds. However, we can prove that (Ext) is compatible with our logic:

Theorem 3.12. *Axiom* (Ext) *is satisfiable.*

Proof. For a given signature, let $(\mathcal{U}, \mathcal{I}_{\mathsf{ty}}, \mathcal{I}, \mathcal{E})$ be an Herbrand interpretation. That is, we define $\mathcal{U}$ to contain the set $\mathcal{U}_\tau$ of all terms of type τ for each ground type τ, we define $\mathcal{I}_{\mathsf{ty}}$ by $\mathcal{I}_{\mathsf{ty}}(\kappa)(\bar{\tau}) = \kappa(\bar{\tau})$, we define $\mathcal{I}$ by $\mathcal{I}(\mathsf{f}, \mathcal{U}_{\bar{\tau}}) = \mathsf{f}\langle\bar{\tau}\rangle$, and we define $\mathcal{E}$ by $\mathcal{E}_{U_\tau, U_\upsilon}(\mathsf{f})(\mathsf{a}) = \mathsf{f}\ \mathsf{a}$. Then $\mathcal{E}_{U_\tau, U_\upsilon}$ is clearly injective and hence $\mathcal{I}$ is extensional. To show that $\mathcal{I} \models$ (Ext), we need to show that (Ext) is true under all valuations. Let ξ be a valuation. If $x \approx y$ is true under ξ, (Ext) is also true. Otherwise $x \approx y$ is false, and hence $\xi(x) \neq \xi(y)$. Then we have $[\![x\ (\mathsf{diff}\langle\alpha, \beta\rangle\ x\ y)]\!]^\xi_{\mathcal{I}} = (\xi(x))\ (\mathsf{diff}\langle\alpha, \beta\rangle\ (\xi(x))\ (\xi(y))) \neq (\xi(y))\ (\mathsf{diff}\langle\alpha, \beta\rangle\ (\xi(x))\ (\xi(y))) = [\![y\ (\mathsf{diff}\langle\alpha, \beta\rangle\ x\ y)]\!]^\xi_{\mathcal{I}}$. Therefore, $x\ (\mathsf{diff}\langle\alpha, \beta\rangle\ x\ y) \not\approx y\ (\mathsf{diff}\langle\alpha, \beta\rangle\ x\ y)$ is true in $\mathcal{I}$ under ξ and so is (Ext). □

3.4. Redundancy Criterion. For our calculi, a redundant (or composite) clause cannot simply be defined as a clause whose ground instances are entailed by smaller ($\prec$) ground instances of existing clauses, because this would make all ArgCong inferences redundant. Our solution is to base the redundancy criterion on a weaker ground logic—ground monomorphic first-order logic—in which argument congruence does not hold. This logic also plays a central role in our refutational completeness proof.

We employ an encoding $\mathcal{F}$ to translate ground λ-free higher-order terms into ground first-order terms. It indexes each symbol occurrence with its type arguments and its term argument count. Thus, $\mathcal{F}(\mathsf{f}) = \mathsf{f}_0$, $\mathcal{F}(\mathsf{f}\ \mathsf{a}) = \mathsf{f}_1(\mathsf{a}_0)$, and $\mathcal{F}(\mathsf{g}\langle\kappa\rangle) = \mathsf{g}_0^\kappa$. This is enough to disable argument congruence; for example, $\{\mathsf{f} \approx \mathsf{h},\ \mathsf{f}\ \mathsf{a} \not\approx \mathsf{h}\ \mathsf{a}\}$ is unsatisfiable, whereas its encoding $\{\mathsf{f}_0 \approx \mathsf{h}_0,\ \mathsf{f}_1(\mathsf{a}_0) \not\approx \mathsf{h}_1(\mathsf{a}_0)\}$ is satisfiable. For clauses built from fully applied ground terms, the two logics are isomorphic, as we would expect from a graceful generalization.

Given a ground λ-free higher-order signature $(\Sigma_{\mathsf{ty}}, \{\}, \Sigma, \{\})$, we define a ground first-order signature $(\Sigma_{\mathsf{ty}}, \{\}, \Sigma_{\mathrm{GF}}, \{\})$ as follows. The type constructors Σ_{ty} are the same in both signatures, but $\to$ is uninterpreted in first-order logic. For each symbol $\mathsf{f} : \Pi\bar{\alpha}_m.\ \tau_1 \to \cdots \to \tau_n \to \tau$ in Σ, where τ is not functional, we introduce symbols $\mathsf{f}_l^{\bar{\upsilon}_m} \in \Sigma_{\mathrm{GF}}$ with argument types $\bar{\tau}_l\sigma$ and return type $(\tau_{l+1} \to \cdots \to \tau_n \to \tau)\sigma$, where $\sigma = \{\bar{\alpha}_m \mapsto \bar{\upsilon}_m\}$, for each tuple of ground types $\bar{\upsilon}_m$ and each $l \in \{0, \ldots, n\}$.

For example, let $\Sigma = \{\mathsf{a} : \kappa,\ \mathsf{g} : \kappa \to \kappa \to \kappa\}$. The corresponding first-order signature is $\Sigma_{\mathrm{GF}} = \{\mathsf{a}_0 : \kappa,\ \mathsf{g}_0 : \kappa \to \kappa \to \kappa,\ \mathsf{g}_1 : \kappa \Rightarrow \kappa \to \kappa,\ \mathsf{g}_2 : \kappa^2 \Rightarrow \kappa\}$ where $\mathsf{f} : \bar{\tau} \Rightarrow \upsilon$ denotes a first-order function symbol f with argument types $\bar{\tau}$ and return type υ, and $\to$ is an uninterpreted binary type constructor. The term $\mathcal{F}(\mathsf{g}\ \mathsf{a}\ \mathsf{a}) = \mathsf{g}_2(\mathsf{a}_0, \mathsf{a}_0)$ has type κ, and $\mathcal{F}(\mathsf{g}\ \mathsf{a}) = \mathsf{g}_1(\mathsf{a}_0)$ has type $\kappa \to \kappa$.

Thus, we consider three layers of logics: the λ-free higher-order layer H over a given signature $(\Sigma_{\mathsf{ty}}, \mathcal{V}_{\mathsf{ty}}, \Sigma, \mathcal{V})$, the ground λ-free higher-order layer GH over the signature $(\Sigma_{\mathsf{ty}}, \{\}, \Sigma, \{\})$, and the ground monomorphic first-order layer GF over the signature $(\Sigma_{\mathsf{ty}}, \{\}, \Sigma_{\mathrm{GF}}, \{\})$ defined above. We use $\mathcal{T}_{\mathrm{H}}$, $\mathcal{T}_{\mathrm{GH}}$, and $\mathcal{T}_{\mathrm{GF}}$ to denote the respective sets of terms, $\mathcal{T}\!\mathit{y}_{\mathrm{H}}$, $\mathcal{T}\!\mathit{y}_{\mathrm{GH}}$, and $\mathcal{T}\!\mathit{y}_{\mathrm{GF}}$ to denote the respective sets of types, and $\mathcal{C}_{\mathrm{H}}$, $\mathcal{C}_{\mathrm{GH}}$, and $\mathcal{C}_{\mathrm{GF}}$ to denote the

respective sets of clauses. In the purifying calculi, we exceptionally let $\mathcal{C}_{\mathrm{H}}$ denote the set of purified clauses. Each of the three layers has an entailment relation $\models$. A clause set N_1 entails a clause set N_2, denoted $N_1 \models N_2$, if any model of N_1 is also a model of N_2. On H and GH, we use λ-free higher-order models for the intensional variants and extensional λ-free higher-order models for the extensional variants; on GF, we use first-order models. This machinery may seem excessive, but it is essential to define redundancy of clauses and inferences properly, and it will play an important role in the refutational completeness proof (Section 4).

The three layers are connected by two functions $\mathcal{G}$ and $\mathcal{F}$. The grounding function $\mathcal{G}$ maps terms from $\mathcal{T}_{\mathrm{H}}$ to the set of their ground instances in $\mathcal{T}_{\mathrm{GH}}$ and clauses from $\mathcal{C}_{\mathrm{H}}$ to the set of their ground instances in $\mathcal{C}_{\mathrm{GH}}$. The encoding $\mathcal{F} : \mathcal{T}_{\mathrm{GH}} \to \mathcal{T}_{\mathrm{GF}}$ is defined recursively as $\mathcal{F}(\mathsf{f}\langle \bar{v}_m \rangle\ \bar{u}) = \mathsf{f}_l^{\bar{v}_m}(\mathcal{F}(\bar{u}_l))$. It can be extended to literals and clauses by mapping each side of a literal and each literal in a clause. The encoding $\mathcal{F}$ is bijective with inverse $\mathcal{F}^{-1}$. Using $\mathcal{F}^{-1}$, the clause order $\succ$ on $\mathcal{T}_{\mathrm{GH}}$ can be transferred to $\mathcal{T}_{\mathrm{GF}}$ by defining $t \succ s$ as equivalent to $\mathcal{F}^{-1}(t) \succ \mathcal{F}^{-1}(s)$. The property that $\succ$ on clauses is the multiset extension of $\succ$ on literals, which in turn is the multiset extension of $\succ$ on terms, is maintained because $\mathcal{F}^{-1}$ maps the multiset representations elementwise.

Crucially, green subterms in $\mathcal{T}_{\mathrm{GH}}$ correspond to subterms in $\mathcal{T}_{\mathrm{GF}}$ (Lemma 3.13), whereas non-green subterms in $\mathcal{T}_{\mathrm{GH}}$ are not subterms at all in $\mathcal{T}_{\mathrm{GF}}$.

To state the correspondence between green subterms in $\mathcal{T}_{\mathrm{GH}}$ and subterms in $\mathcal{T}_{\mathrm{GF}}$ explicitly, we define positions of green subterms as follows. A term $s \in \mathcal{T}_{\mathrm{H}}$ is a green subterm at position ϵ of s. If a term $s \in \mathcal{T}_{\mathrm{H}}$ is a green subterm at position p of u_i for some $1 \leq i \leq n$, then s is a green subterm at position $i.p$ of $\mathsf{f}\langle \bar{\tau} \rangle\ \bar{u}_n$ and of $x\ \bar{u}_n$. For $\mathcal{T}_{\mathrm{GF}}$, positions are defined as usual in first-order logic.

Lemma 3.13. *Let $s, t \in \mathcal{T}_{\mathrm{GH}}$. We have $\mathcal{F}(t\langle s \rangle_p) = \mathcal{F}(t)[\mathcal{F}(s)]_p$. In other words, s is a green subterm of t at position p if and only if $\mathcal{F}(s)$ is a subterm of $\mathcal{F}(t)$ at position p.*

Proof. By induction on p. If $p = \varepsilon$, then $s = t[s]_p$. Hence $\mathcal{F}(t[s]_p) = \mathcal{F}(s) = \mathcal{F}(t)\langle \mathcal{F}(s) \rangle_p$. If $p = i.p'$, then $t[s]_p = \mathsf{f}\langle \bar{\tau} \rangle\ \bar{u}_n$ with $u_i = u_i[s]_{p'}$. Applying $\mathcal{F}$, we obtain by the induction hypothesis that $\mathcal{F}(u_i) = \mathcal{F}(u_i)\langle \mathcal{F}(s) \rangle_{p'}$. Therefore, $\mathcal{F}(t[s]_p) = \mathsf{f}_n^{\bar{\tau}}(\mathcal{F}(u_1), \ldots, \mathcal{F}(u_{i-1}), \mathcal{F}(u_i)\langle \mathcal{F}(s) \rangle_{p'}, \mathcal{F}(u_{i+1}), \ldots, \mathcal{F}(u_n))$. It follows that $\mathcal{F}(t[s]_p) = \mathcal{F}(t)\langle \mathcal{F}(s) \rangle_p$. □

Corollary 3.14. *Given $s, t \in \mathcal{T}_{\mathrm{GF}}$, we have $\mathcal{F}^{-1}(t[s]_p) = \mathcal{F}^{-1}(t)\langle \mathcal{F}^{-1}(s) \rangle_p$.*

Lemma 3.15. *Well-foundedness, totality, compatibility with contexts, and the subterm property hold for $\succ$ on $\mathcal{T}_{\mathrm{GF}}$.*

Proof. Compatibility with contexts: We must show that $s \succ s'$ implies $t[s]_p \succ t[s']_p$ for terms $t, s, s' \in \mathcal{T}_{\mathrm{GF}}$. Assuming $s \succ s'$, we have $\mathcal{F}^{-1}(s) \succ \mathcal{F}^{-1}(s')$. By compatibility with green contexts on $\mathcal{T}_{\mathrm{GH}}$, we have $\mathcal{F}^{-1}(t)\langle \mathcal{F}^{-1}(s) \rangle_p \succ \mathcal{F}^{-1}(t)\langle \mathcal{F}^{-1}(s') \rangle_p$. By Corollary 3.14, we have $t[s]_p \succ t[s']_p$.

Well-foundedness: Assume that there exists an infinite descending chain $t_1 \succ t_2 \succ \cdots$ in $\mathcal{T}_{\mathrm{GF}}$. By applying $\mathcal{F}^{-1}$, we then obtain the chain $\mathcal{F}^{-1}(t_1) \succ \mathcal{F}^{-1}(t_2) \succ \cdots$ in $\mathcal{T}_{\mathrm{GH}}$, contradicting well-foundedness on $\mathcal{T}_{\mathrm{GH}}$.

Totality: Let $s, t \in \mathcal{T}_{\mathrm{GF}}$. Then $\mathcal{F}^{-1}(t)$ and $\mathcal{F}^{-1}(s)$ must be comparable by totality in $\mathcal{T}_{\mathrm{GH}}$. Hence, t and s are comparable.

SUBTERM PROPERTY: By Corollary 3.14 and the subterm property on $\mathcal{T}_{\mathrm{GH}}$, $\mathcal{F}^{-1}(t[s]_p) = \mathcal{F}^{-1}(t)\langle\mathcal{F}^{-1}(s)\rangle_p \succ \mathcal{F}^{-1}(s)$. Hence, $t[s]_p \succ s$. □

In standard superposition, redundancy relies on the entailment relation $\models$ on ground clauses. We will define redundancy on GH and H in the same way, but using GF's entailment relation. This notion of redundancy gracefully generalizes the first-order notion, without making all ArgCong inferences redundant.

The standard redundancy criterion for standard superposition cannot justify subsumption deletion. Following Waldmann et al. [66], we define a redundancy criterion that can justify subsumption deletion by employing a well-founded order $\sqsupset$ on $\mathcal{C}_{\mathrm{H}}$.

We define the sets of redundant clauses w.r.t. a given clause set as follows:

- Given $C \in \mathcal{C}_{\mathrm{GF}}$ and $N \subseteq \mathcal{C}_{\mathrm{GF}}$, let $C \in \mathit{GFRed}_{\mathrm{C}}(N)$ if $\{D \in N \mid D \prec C\} \models C$.
- Given $C \in \mathcal{C}_{\mathrm{GH}}$ and $N \subseteq \mathcal{C}_{\mathrm{GH}}$, let $C \in \mathit{GHRed}_{\mathrm{C}}(N)$ if $\mathcal{F}(C) \in \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N))$.
- Given $C \in \mathcal{C}_{\mathrm{H}}$ and $N \subseteq \mathcal{C}_{\mathrm{H}}$, let $C \in \mathit{HRed}_{\mathrm{C}}(N)$ if for every $D \in \mathcal{G}(C)$, we have $D \in \mathit{GHRed}_{\mathrm{C}}(\mathcal{G}(N))$ or there exists $C' \in N$ such that $C \sqsupset C'$ and $D \in \mathcal{G}(C)$.

Along with the three layers of logics, we consider three inference systems: *HInf*, *GHInf* and *GFInf*. *HInf* is one of the four variants of the inference system described in Section 3.1. For uniformity, we regard axiom (Ext) as a premise-free inference rule Ext whose conclusion is (Ext). In the purifying calculi, the conclusion of Ext must be purified. *GHInf* consists of all Sup, EqRes, and EqFact inferences from *HInf* whose premises and conclusion are ground, a premise-free rule GExt whose infinitely many conclusions are the ground instances of (Ext), and the following ground variant of ArgCong:

$$\frac{C' \vee s \approx s'}{C' \vee s\,\bar{u}_n \approx s'\,\bar{u}_n}\,\textsc{GArgCong}$$

where $s \approx s'$ is strictly eligible in the premise and $\bar{u}_n$ is a nonempty tuple of ground terms. *GFInf* contains all Sup, EqRes, and EqFact inferences from *GHInf* translated by $\mathcal{F}$. It coincides exactly with standard first-order superposition. Given a Sup, EqRes, or EqFact inference $\iota \in \mathit{GHInf}$, let $\mathcal{F}(\iota)$ denote the corresponding inference in *GFInf*.

Given an inference ι, we write $\mathit{prems}(\iota)$ for the tuple of premises, $\mathit{mprem}(\iota)$ for the main (i.e., rightmost) premise, $\mathit{preconcl}(\iota)$ for the conclusion before purification, and $\mathit{concl}(\iota)$ for the conclusion after purification. For the nonpurifying variants, $\mathit{preconcl}(\iota) = \mathit{concl}(\iota)$ simply denotes the conclusion.

Each of the three inference systems is parameterized by a selection function *sel*. Occasionally, we will make this dependency explicit, writing $\mathit{HInf}^{\mathit{sel}}$ for *HInf* and similarly for *GHInf* and *GFInf*. For each selection function *sel* on $\mathcal{C}_{\mathrm{GH}}$, via the bijection $\mathcal{F}$, we can obtain a corresponding selection function on $\mathcal{C}_{\mathrm{GF}}$, which we denote $\mathcal{F}(\mathit{sel})$. There is, however, no general way to derive the right selection function on $\mathcal{C}_{\mathrm{GH}}$ from a selection function on $\mathcal{C}_{\mathrm{H}}$. In the refutational completeness proof, given a saturated clause set $N \subseteq \mathcal{C}_{\mathrm{H}}$ and a selection function on $\mathcal{C}_{\mathrm{H}}$, we need a selection function on $\mathcal{C}_{\mathrm{GH}}$ such that for each clause $C \in \mathcal{G}(N)$ there exists a clause $D \in N$ with $C \in \mathcal{G}(D)$ and corresponding selected literals. Since the saturated clause set N is not known during a derivation, our redundancy criterion may not depend on it. Therefore, we consider all selection functions on $\mathcal{C}_{\mathrm{GH}}$ such that for each clause in $C \in \mathcal{C}_{\mathrm{GH}}$, there exists a clause $D \in \mathcal{C}_{\mathrm{H}}$ with $C \in \mathcal{G}(D)$ and corresponding selected literals. Given a selection function *sel* on $\mathcal{C}_{\mathrm{H}}$, let $\mathcal{G}(\mathit{sel})$ denote the set of such selection functions on $\mathcal{C}_{\mathrm{GH}}$.

Given a selection function *sel* on $\mathcal{C}_{\mathrm{H}}$, a selection function $\mathit{gsel} \in \mathcal{G}(\mathit{sel})$, and a non-PosExt inference $\iota \in \mathit{HInf}^{\mathit{sel}}$, we define the set $\mathcal{G}^{\mathit{gsel}}(\iota)$ of ground instances of ι to be all

inferences $\iota' \in \mathit{GHInf}^{\mathit{gsel}}$ such that $\mathit{prems}(\iota') = \mathit{prems}(\iota)\theta$ and $\mathit{concl}(\iota') = \mathit{preconcl}(\iota)\theta$ for some grounding substitution θ. This will map Sup to Sup, EqFact to EqFact, EqRes to EqRes, Ext to GExt, and ArgCong to GArgCong inferences, but it is also possible that $\mathcal{G}^{\mathit{gsel}}(\iota)$ is the empty set for some inferences ι. For PosExt inferences ι, which cannot be grounded, we let $\mathcal{G}^{\mathit{gsel}}(\iota) = \mathit{undef}$.

We define the sets of redundant inferences w.r.t. a given clause set as follows:

- Given $\iota \in \mathit{GFInf}^{\mathit{gsel}}$ and $N \subseteq \mathcal{C}_{\mathrm{GF}}$, let $\iota \in \mathit{GFRed}_{\mathrm{I}}^{\mathit{gsel}}(N)$ if $\mathit{prems}(\iota) \cap \mathit{GFRed}_{\mathrm{C}}(N) \neq \varnothing$ or $\{D \in N \mid D \prec \mathit{mprem}(\iota)\} \models \mathit{concl}(\iota)$.
- Given $\iota \in \mathit{GHInf}^{\mathit{gsel}}$ and $N \subseteq \mathcal{C}_{\mathrm{GH}}$, let $\iota \in \mathit{GHRed}_{\mathrm{I}}^{\mathit{gsel}}(N)$ if
  - ι is not a GArgCong or GExt inference and $\mathcal{F}(\iota) \in \mathit{GFRed}_{\mathrm{I}}^{\mathcal{F}(\mathit{gsel})}(\mathcal{F}(N))$; or
  - the calculus variant is nonpurifying and ι is a GArgCong or GExt inference and $\mathit{concl}(\iota) \in N \cup \mathit{GHRed}_{\mathrm{C}}(N)$; or
  - the calculus variant is purifying and ι is a GArgCong or GExt inference and $\mathcal{F}(N) \models \mathcal{F}(\mathit{concl}(\iota))$.
- Given $\iota \in \mathit{HInf}^{\mathit{sel}}$ and $N \subseteq \mathcal{C}_{\mathrm{H}}$, let $\iota \in \mathit{HRed}_{\mathrm{I}}(N)$ if
  - ι is not a PosExt inference and $\mathcal{G}^{\mathit{gsel}}(\iota) \subseteq \mathit{GHRed}_{\mathrm{I}}(\mathcal{G}(N))$ for all $\mathit{gsel} \in \mathcal{G}(\mathit{sel})$; or
  - ι is a PosExt inference and $\mathcal{G}(\mathit{concl}(\iota)) \in \mathcal{G}(N) \cup \mathit{GHRed}_{\mathrm{C}}(\mathcal{G}(N))$.

Occasionally, we omit the selection function in the notation when it is irrelevant. A clause set N is *saturated* w.r.t. an inference system and a redundancy criterion $(\mathit{Red}_{\mathrm{I}}, \mathit{Red}_{\mathrm{C}})$ if every inference from clauses in N is in $\mathit{Red}_{\mathrm{I}}(N)$.

4. Refutational Completeness

Besides soundness, the most important property of the the four calculi introduced in Section 3.1 is refutational completeness. To circumvent the term order's potential nonmonotonicity, our Sup inference rule only considers green subterms. This is reflected in our proof by the reliance on the layer GF introduced in Section 3.4. The equation $\mathsf{g}_0 \approx \mathsf{f}_0 \in \mathcal{C}_{\mathrm{GF}}$, which corresponds to the equation $\mathsf{g} \approx \mathsf{f} \in \mathcal{C}_{\mathrm{GH}}$, cannot be used directly to rewrite the clause $\mathsf{g}_1(\mathsf{a}_0) \not\approx \mathsf{f}_1(\mathsf{a}_0) \in \mathcal{C}_{\mathrm{GF}}$, which corresponds to $\mathsf{g}\ \mathsf{a} \not\approx \mathsf{f}\ \mathsf{a} \in \mathcal{C}_{\mathrm{GH}}$. Instead, we first need to apply ArgCong to derive $\mathsf{g}\ x \approx \mathsf{f}\ x$, which after grounding and transfer to GF yields $\mathsf{g}_1(\mathsf{a}_0) \approx \mathsf{f}_1(\mathsf{a}_0)$. The GF layer is a device that enables us to reuse the refutational completeness result for standard (first-order) superposition.

The proof proceeds in three steps, corresponding to the three layers GF, GH, and H introduced in Section 3.4:

(1) We use Bachmair and Ganzinger's work on the refutational completeness of standard superposition [4] to prove static refutational completeness of *GFInf*.
(2) From the first-order model constructed in Bachmair and Ganzinger's proof, we derive a clausal λ-free higher-order model to prove static refutational completeness of *GHInf*.
(3) We use the saturation framework of Waldmann et al. [66] to lift the static refutational completeness of *GHInf* to static and dynamic refutational completeness of *HInf*.

In step (1), since the inference system *GFInf* is standard ground superposition, we only need to work around minor differences between Bachmair and Ganzinger's definitions and ours. Given a saturated clause set $N \subseteq \mathcal{C}_{\mathrm{GF}}$ with $\bot \notin N$, Bachmair and Ganzinger prove refutational completeness by constructing a term rewriting system R_N and showing that it can be viewed as an interpretation that is a model of N. This step is exclusively concerned with ground first-order clauses.

In step (2), we derive refutational completeness of *GHInf*. Given a saturated clause set $N \subseteq \mathcal{C}_{\mathrm{GH}}$ with $\bot \notin N$, we use the first-order model $R_{\mathcal{F}(N)}$ of $\mathcal{F}(N)$ constructed in step (1) to derive a clausal higher-order interpretation that is a model of N. Thanks to saturation w.r.t. GArgCong, the higher-order interpretation can conflate the interpretations of the members $\mathsf{f}_0^{\bar{\upsilon}}, \ldots, \mathsf{f}_n^{\bar{\upsilon}}$ of a same symbol family. In the extensional variants, saturation w.r.t. GExt can be used to show that the constructed interpretation is extensional.

In step (3), we employ the saturation framework of Waldmann et al. [66] to prove refutational completeness of *HInf*. The main proof obligation the framework leaves to us is that there exist inferences in *HInf* corresponding to all nonredundant inferences in *GHInf*. For monotone term orders, we can avoid Sup inferences into variables x by exploiting the clause order's compatibility with contexts: If $t' \prec t$, we have $C\{x \mapsto t'\} \prec C\{x \mapsto t\}$, which allows us to show that Sup inferences into variables are redundant. This technique fails for variables x that occur applied in C, because the order lacks compatibility with arguments. This is why the calculi must either purify clauses to make this line of reasoning work again or perform some Sup inferences into variables.

4.1. The Ground First-Order Layer. We use Bachmair and Ganzinger's results on standard superposition [4] to prove refutational completeness of GF. In the subsequent steps, we will also make use of specific properties of Bachmair and Ganzinger's model.

Bachmair and Ganzinger's logic and inference system differ in some details from GF:

- In addition to their side conditions on the equality factoring rule, our rule EqFact requires that $s'\sigma \not\preceq t'\sigma$. Bachmair and Ganzinger's proofs can be easily adapted to cope with this additional restriction. In the proof of their Lemma 4.11, case (iii), we have $s \neq t$ because $s \succ t$ by assumption. In the proof of the same lemma, case (iv), we have $u \neq v$ because $\Delta \cap I_C = \emptyset$.
- Bachmair and Ganzinger use untyped first-order logic, whereas GF's logic is typed. Bachmair and Ganzinger's proof works verbatim for monomorphic first-order logic as well, but we need to require that the order $\succ$ has the subterm property to show that there exist no critical pairs in the term rewriting system, as observed by Wand [67, Section 3.2.1].
- In their redundancy criterion for clauses, Bachmair and Ganzinger require that a finite subset of $\{D \in N \mid D \prec C\}$ entails C, whereas we require that $\{D \in N \mid D \prec C\}$ entails C. By compactness of first-order logic, the two criteria are equivalent.

The basis of Bachmair and Ganzinger's proof is that a term rewriting system R defines an interpretation $\mathcal{T}_{\mathrm{GF}}/R$ such that for every ground equation $s \approx t$, we have $\mathcal{T}_{\mathrm{GF}}/R \models s \approx t$ if and only if $s \leftrightarrow_R^* t$. Formally, $\mathcal{T}_{\mathrm{GF}}/R$ denotes the monomorphic first-order interpretation whose universes U_τ consist of the R-equivalence classes over $\mathcal{T}_{\mathrm{GF}}$ containing terms of type τ. The interpretation $\mathcal{T}_{\mathrm{GF}}/R$ is term-generated—that is, for every element a of the universe of this interpretation and for any valuation ξ, there exists a ground term t such that $[\![t]\!]^\xi_{\mathcal{T}_{\mathrm{GF}}/R} = a$. To lighten notation, we will write R to refer to both the term rewriting system R and the interpretation $\mathcal{T}_{\mathrm{GF}}/R$.

The term rewriting system is constructed as follows. Let $N \subseteq \mathcal{C}_{\mathrm{GF}}$. We first define sets of rewrite rules E_N^C and R_N^C for all $C \in N$ by induction on the clause order. Assume that E_N^D has already been defined for all $D \in N$ such that $D \prec C$. Then $R_N^C = \bigcup_{D \prec C} E_N^D$. Let $E_N^C = \{s \to t\}$ if the following conditions are met:

(a) $C = C' \vee s \approx t$;
(b) $s \approx t$ is strictly maximal in C;

(c) $s \succ t$;
(d) C' is false in R_N^C;
(e) s is irreducible w.r.t. R_N^C.

Then C is said to *produce* $s \to t$. Otherwise, $E_N^C = \emptyset$. Finally, $R_N = \bigcup_D E_N^D$.

We call an inference $\iota \in \mathit{GFInf}$ *B&G-redundant* if some $C \in \mathit{prems}(\iota)$ is true in R_N^C or $\mathit{concl}(\iota)$ is true in $R_N^{\mathit{mprem}(\iota)}$. We call a set $N \subseteq \mathcal{C}_{\mathrm{GF}}$ *B&G-saturated* if all inferences from N are B&G-redundant.

Lemma 4.1. *If $\bot \notin N$ and $N \subseteq \mathcal{C}_{\mathrm{GF}}$ is saturated w.r.t. GFInf and $\mathit{GFRed}_{\mathrm{I}}$, then N is B&G-saturated.*

Proof. Let $N \subseteq \mathcal{C}_{\mathrm{GF}}$ be saturated w.r.t. *GFInf* and $\mathit{GFRed}_{\mathrm{I}}$. To show that N is B&G-saturated, let ι be an inference from N. We need to show that ι is B&G-redundant w.r.t. N. We proceed by well-founded induction on $\mathit{mprem}(\iota)$ w.r.t. $\succ$. By the induction hypothesis, for all inferences ι' with $\mathit{concl}(\iota') \prec \mathit{mprem}(\iota)$, ι' is B&G-redundant w.r.t. N. By Lemma 5.5 of Bachmair and Ganzinger, ι is B&G-redundant w.r.t. N. □

Lemma 4.2. *Let $\bot \notin N$ and $N \subseteq \mathcal{C}_{\mathrm{GF}}$ be saturated w.r.t. GFInf and $\mathit{GFRed}_{\mathrm{I}}$. If $C = C' \vee s \approx t \in N$ produces $s \to t$, then $s \approx t$ is strictly eligible in C and C' is false in R_N.*

Proof. By Lemma 4.1, N is also B&G-saturated. By condition (d), C' is false in R_N^C. Since $s \succ t$ by condition (c) and s is irreducible w.r.t. R_N^C by condition (e), $s \approx t$ is also false in R_N^C. Hence, C is false in R_N^C. Using this and conditions (a), (b), (c), and (e), we can apply Lemma 4.11 of Bachmair and Ganzinger. Part (ii) of that lemma shows that $s \approx t$ is strictly eligible in C, and part (iv) shows that C' is false in R_N. □

Theorem 4.3 (Ground first-order static refutational completeness)**.** *Let $\bot \notin N$ and $N \subseteq \mathcal{C}_{\mathrm{GF}}$ be saturated w.r.t. GFInf and $\mathit{GFRed}_{\mathrm{I}}$. Then R_N is a model of N.*

Proof. By Lemma 4.1, N is also B&G-saturated. It follows that R_N is a model of N, as shown in the proof of Theorem 4.14 of Bachmair and Ganzinger. □

4.2. The Ground Higher-Order Layer. In this subsection, let *sel* be a selection function on $\mathcal{C}_{\mathrm{GH}}$, let $N \subseteq \mathcal{C}_{\mathrm{GH}}$ be a clause set saturated w.r.t. $\mathit{GHInf}^{\mathit{sel}}$ and $\mathit{GHRed}_{\mathrm{I}}^{\mathit{sel}}$, and let $\bot \notin N$. Clearly, $\mathcal{F}(N)$ is then saturated w.r.t. $\mathit{GFInf}^{\mathcal{F}(\mathit{sel})}$ and $\mathit{GFRed}_{\mathrm{I}}^{\mathcal{F}(\mathit{sel})}$.

We abbreviate $R_{\mathcal{F}(N)}$ as R. From R, we construct a model $\mathcal{I}^{\mathrm{GH}}$ of N. The key properties enabling us to perform this construction are that R is term-generated and that the interpretations of the members $\mathsf{f}_0^{\bar{\upsilon}}, \ldots, \mathsf{f}_n^{\bar{\upsilon}}$ of a same symbol family behave in the same way thanks to the ArgCong rule.

Lemma 4.4 (Argument congruence)**.** *For terms $s, t, u \in \mathcal{T}_{\mathrm{GH}}$, if $[\![\mathcal{F}(s)]\!]_R = [\![\mathcal{F}(t)]\!]_R$, then $[\![\mathcal{F}(s\,u)]\!]_R = [\![\mathcal{F}(t\,u)]\!]_R$.*

Proof. What we want to show is equivalent to

$$\mathcal{F}(s) \leftrightarrow_R^* \mathcal{F}(t) \text{ implies } \mathcal{F}(s\,u) \leftrightarrow_R^* \mathcal{F}(t\,u)$$

By induction on the number of rewrite steps and due to symmetry, it suffices to show that

$$\mathcal{F}(s) \to_R \mathcal{F}(t) \text{ implies } \mathcal{F}(s\,u) \leftrightarrow_R^* \mathcal{F}(t\,u)$$

If the rewrite step from $\mathcal{F}(s)$ is below the top level, this is obvious because there is an corresponding rewrite step from $\mathcal{F}(s\,u)$. If it is at the top level, $\mathcal{F}(s) \to \mathcal{F}(t)$ must be rule

of R. This rule must come from a productive clause of the form $\mathcal{F}(C) = \mathcal{F}(C' \vee s \approx t)$. By Lemma 4.2, $\mathcal{F}(s \approx t)$ is strictly eligible in $\mathcal{F}(C)$ w.r.t. $\mathcal{F}(\mathit{sel})$, and hence $s \approx t$ is strictly eligible in C w.r.t. sel. Moreover, s and t have functional type. Thus, the following GArgCong inference ι is applicable:

$$\frac{C' \vee s \approx t}{C' \vee s\, u \approx t\, u}\,\textsc{GArgCong}$$

By saturation, ι is redundant w.r.t. N—i.e., we have $\mathit{concl}(\iota) \in N \cup \mathit{GHRed}_{\mathrm{C}}(N)$ (for nonpurifying variants) or $\mathcal{F}(N) \models \mathit{concl}(\iota)$ (for purifying variants). In both cases, by Theorem 4.3, $\mathcal{F}(\mathit{concl}(\iota))$ is then true in R. By Lemma 4.2, $\mathcal{F}(C')$ is false in R. Therefore, $\mathcal{F}(s\, u \approx t\, u)$ must be true in R. □

Lemma 4.5. *For terms $s, t, u, v \in \mathcal{T}_{\mathrm{GH}}$, if $[\![\mathcal{F}(s)]\!]_R = [\![\mathcal{F}(t)]\!]_R$ and $[\![\mathcal{F}(u)]\!]_R = [\![\mathcal{F}(v)]\!]_R$, then $[\![\mathcal{F}(s\, u)]\!]_R = [\![\mathcal{F}(t\, v)]\!]_R$.*

Proof. By Lemma 4.4, we have $[\![\mathcal{F}(s\, u)]\!]_R = [\![\mathcal{F}(t\, u)]\!]_R$. It remains to show that $[\![\mathcal{F}(t\, u)]\!]_R = [\![\mathcal{F}(t\, v)]\!]_R$. Since t is ground, it must be of the form $\mathsf{f}\langle \bar{\upsilon}_m \rangle\, \bar{t}_n$. Let $\mathcal{J}$ be the interpretation function of the interpretation R. Then

$$[\![\mathcal{F}(t\, u)]\!]_R = \mathcal{J}(\mathsf{f}_{n+1}^{\bar{\upsilon}_m})([\![\mathcal{F}(\bar{t}_n)]\!]_R, [\![\mathcal{F}(u)]\!]_R) = \mathcal{J}(\mathsf{f}_{n+1}^{\bar{\upsilon}_m})([\![\mathcal{F}(\bar{t}_n)]\!]_R, [\![\mathcal{F}(v)]\!]_R) = [\![\mathcal{F}(t\, v)]\!]_R \qquad \square$$

Definition 4.6. Define a higher-order interpretation $\mathcal{I}^{\mathrm{GH}} = (\mathcal{U}^{\mathrm{GH}}, \mathcal{J}_{\mathsf{ty}}^{\mathrm{GH}}, \mathcal{J}^{\mathrm{GH}}, \mathcal{E}^{\mathrm{GH}})$ as follows. The interpretation R defined above is an interpretation in monomorphic first-order logic. Let U_τ be its universe for type τ and and $\mathcal{J}$ its interpretation function. Let $\mathcal{U}^{\mathrm{GH}} = \{U_\tau \mid \tau \text{ is a ground type}\}$. Let $\mathcal{J}_{\mathsf{ty}}^{\mathrm{GH}}(\kappa)(U_{\bar{\tau}}) = U_{\kappa(\bar{\tau})}$ for all type constructors κ and type tuples $\bar{\tau}$. Let $\mathcal{J}^{\mathrm{GH}}(\mathsf{f}, U_{\bar{\tau}}) = \mathcal{J}(\mathsf{f}_0^{\bar{\tau}})$.

Since R is term-generated, for every $a \in U_{\tau\to\upsilon}$ and $b \in U_\tau$, there exist ground terms $s : \tau \to \upsilon$ and $u : \tau$ such that $[\![\mathcal{F}(s)]\!]_R = a$ and $[\![\mathcal{F}(u)]\!]_R = b$. We define $\mathcal{E}^{\mathrm{GH}}$ by $\mathcal{E}^{\mathrm{GH}}_{U_\tau, U_\upsilon}(a)(b) = [\![\mathcal{F}(s\, u)]\!]_R$ for any term u. By Lemma 4.5, this definition is independent of the choice of s and u.

Lemma 4.7 (Model transfer to GH)**.** *$\mathcal{I}^{\mathrm{GH}}$ is a model of N. In the extensional variants, $\mathcal{I}^{\mathrm{GH}}$ is an extensional model of N.*

Proof. We first prove by induction on terms $t \in \mathcal{T}_{\mathrm{GH}}$ that $[\![t]\!]_{\mathcal{I}^{\mathrm{GH}}} = [\![\mathcal{F}(t)]\!]_R$. Let $t \in \mathcal{T}_{\mathrm{GH}}$, and assume as the induction hypothesis that $[\![u]\!]_{\mathcal{I}^{\mathrm{GH}}} = [\![\mathcal{F}(u)]\!]_R$ for all subterms u of t. If t is of the form $\mathsf{f}\langle \bar{\upsilon} \rangle$, then

$$[\![t]\!]_{\mathcal{I}^{\mathrm{GH}}} = \mathcal{J}^{\mathrm{GH}}(\mathsf{f}, U_{\bar{\upsilon}}) = \mathcal{J}(\mathsf{f}_0^{\bar{\upsilon}}) = [\![\mathsf{f}_0^{\bar{\upsilon}}]\!]_R = [\![\mathcal{F}(\mathsf{f}\langle \bar{\upsilon} \rangle)]\!]_R = [\![\mathcal{F}(t)]\!]_R$$

If t is an application $t = t_1\, t_2$, where t_1 is of type $\tau \to \upsilon$, then, using the definition of the term denotation and of $\mathcal{E}^{\mathrm{GH}}$, we have

$$[\![t_1\, t_2]\!]_{\mathcal{I}^{\mathrm{GH}}} = \mathcal{E}^{\mathrm{GH}}_{U_\tau, U_\upsilon}([\![t_1]\!]_{\mathcal{I}^{\mathrm{GH}}})([\![t_2]\!]_{\mathcal{I}^{\mathrm{GH}}}) \overset{\mathrm{IH}}{=} \mathcal{E}^{\mathrm{GH}}_{U_\tau, U_\upsilon}([\![\mathcal{F}(t_1)]\!]_R)([\![\mathcal{F}(t_2)]\!]_R) = [\![\mathcal{F}(t_1\, t_2)]\!]_R$$

So we have shown that $[\![t]\!]_{\mathcal{I}^{\mathrm{GH}}} = [\![\mathcal{F}(t)]\!]_R$ for all terms t. It follows that a ground equation $s \approx t$ is true in $\mathcal{I}^{\mathrm{GH}}$ if and only if $\mathcal{F}(s \approx t)$ is true in R. Hence a ground clause C is true in $\mathcal{I}^{\mathrm{GH}}$ if and only if $\mathcal{F}(C)$ is true in R. By Theorem 4.3, R is a model of $\mathcal{F}(N)$. Thus, $\mathcal{I}^{\mathrm{GH}}$ is a model of N.

For the extensional variants, it remains to show that $\mathcal{I}^{\mathrm{GH}}$ is extensional—i.e., we have to show that for all τ and υ and all $a, b \in U_{\tau\to\upsilon}$, if $a \neq b$, then $\mathcal{E}^{\mathrm{GH}}(a) \neq \mathcal{E}^{\mathrm{GH}}(b)$. Since R is term-generated, there are terms $s, t \in \mathcal{T}_{\mathrm{GF}}$ such that $[\![s]\!]_R = a$ and $[\![t]\!]_R = b$. By what

we have shown above, it follows that $[\![s']\!]_{\mathcal{I}^{\mathrm{GH}}} = a$ and $[\![t']\!]_{\mathcal{I}^{\mathrm{GH}}} = b$ for $s' = \mathcal{F}^{-1}(s)$ and $t' = \mathcal{F}^{-1}(t)$.

Since N is saturated, the GExt inference that generates the clause

$$C \;=\; s'\ (\mathsf{diff}\langle \tau, \upsilon\rangle\ s'\ t') \not\approx t'\ (\mathsf{diff}\langle \tau, \upsilon\rangle\ s'\ t') \vee s' \approx t'$$

is redundant—i.e., $C \in N \cup \mathit{GHRed}_{\mathrm{C}}(N)$ (in nonpurifying variants) or $\mathcal{F}(N) \models \mathcal{F}(C)$ (in purifying variants). In both cases, it follows that $R \models \mathcal{F}(C)$ by Theorem 4.3 and thus $\mathcal{I}^{\mathrm{GH}} \models C$ by what we have shown above. The second literal of C is false in $\mathcal{I}^{\mathrm{GH}}$ because $[\![s']\!]_{\mathcal{I}^{\mathrm{GH}}} = a \neq b = [\![t']\!]_{\mathcal{I}^{\mathrm{GH}}}$. So the first literal of C must be true in $\mathcal{I}^{\mathrm{GH}}$ and thus

$$\begin{aligned}\mathcal{E}^{\mathrm{GH}}(a)([\![\mathsf{diff}\langle \tau, \upsilon\rangle\ s'\ t']\!]_{\mathcal{I}^{\mathrm{GH}}}) &= [\![s'\ (\mathsf{diff}\langle \tau, \upsilon\rangle\ s'\ t')]\!]_{\mathcal{I}^{\mathrm{GH}}} \\ &\neq [\![t'\ (\mathsf{diff}\langle \tau, \upsilon\rangle\ s'\ t')]\!]_{\mathcal{I}^{\mathrm{GH}}} = \mathcal{E}^{\mathrm{GH}}(b)([\![\mathsf{diff}\langle \tau, \upsilon\rangle\ s'\ t']\!]_{\mathcal{I}^{\mathrm{GH}}})\end{aligned}$$

It follows that $\mathcal{E}^{\mathrm{GH}}(a) \neq \mathcal{E}^{\mathrm{GH}}(b)$. □

We summarize the results of this subsection in the following theorem:

Theorem 4.8 (Ground static refutational completeness)**.** *Let $N \subseteq \mathcal{C}_{\mathrm{GH}}$ be a clause set saturated w.r.t. $\mathit{GHInf}^{\mathit{gsel}}$ and $\mathit{GHRed}_{\mathrm{I}}^{\mathit{gsel}}$ for some selection function gsel. Then $N \models \bot$ if and only if $\bot \in N$.*

4.3. The Nonground Higher-Order Layer. To lift the result to the nonground layer, we employ the saturation framework of Waldmann et al. [66]. It is easy to see that the entailment relation $\models$ on GH is a consequence relation in the sense of the framework. It remains to show that our redundancy criterion on GH is a redundancy criterion in the sense of the framework and that $\mathcal{G}$ is a grounding function in the sense of the framework:

Lemma 4.9. *Given an interpretation $\mathcal{I}$ on* GH*, there exists an interpretation $\mathcal{I}^{\mathrm{GF}}$ on* GF *such that for any clause $C \in \mathcal{C}_{\mathrm{GH}}$ the truth values of C in $\mathcal{I}$ and of $\mathcal{F}(C)$ in $\mathcal{I}^{\mathrm{GF}}$ coincide.*

Proof. Let $\mathcal{I} = (\mathcal{U}, \mathcal{J}_{\mathsf{ty}}, \mathcal{J}, \mathcal{E})$ be an interpretation on GH. Let $\mathcal{U}_\tau^{\mathrm{GF}} = [\![\tau]\!]_{\mathcal{I}_{\mathsf{ty}}}$ be the first-order type universe for the ground type τ. For a symbol $\mathsf{f}_l^{\bar{\upsilon}_m} \in \Sigma_{\mathrm{GF}}$ and universe elements $\bar{a}_l$, let $\mathcal{J}^{\mathrm{GF}}(\mathsf{f}_l^{\bar{\upsilon}_m})(\bar{a}_l) = [\![\mathsf{f}\langle \bar{\upsilon}_m\rangle\ \bar{x}_l]\!]_{\mathcal{I}}^{\{\bar{x}_l \mapsto \bar{a}\}}$. This defines an interpretation $\mathcal{I}^{\mathrm{GF}} = (\mathcal{U}^{\mathrm{GF}}, \mathcal{J}^{\mathrm{GF}})$ on GF.

We need to show that for any $C \in \mathcal{C}_{\mathrm{GH}}$, $\mathcal{I} \models C$ if and only if $\mathcal{I}^{\mathrm{GF}} \models \mathcal{F}(C)$. It suffices to show that $[\![t]\!]_{\mathcal{I}} = [\![\mathcal{F}(t)]\!]_{\mathcal{I}^{\mathrm{GF}}}$ for all terms $t \in \mathcal{T}_{\mathrm{GH}}$. We prove this by induction on t. Since t is ground, it must be of the form $\mathsf{f}\langle \bar{\upsilon}_m\rangle\ \bar{s}_l$. Then $\mathcal{F}(t) = \mathsf{f}_l^{\bar{\upsilon}_m}(\mathcal{F}(\bar{s}_l))$ and hence

$$[\![\mathcal{F}(t)]\!]_{\mathcal{I}^{\mathrm{GF}}} = \mathcal{J}^{\mathrm{GF}}(\mathsf{f}_l^{\bar{\upsilon}_m})([\![\mathcal{F}(\bar{s}_l)]\!]_{\mathcal{I}^{\mathrm{GF}}}) \overset{\mathrm{IH}}{=} \mathcal{J}^{\mathrm{GF}}(\mathsf{f}_l^{\bar{\upsilon}_m})([\![\bar{s}_l]\!]_{\mathcal{I}}) = [\![\mathsf{f}\langle \bar{\upsilon}_m\rangle\ \bar{s}_l]\!]_{\mathcal{I}} = [\![t]\!]_{\mathcal{I}}$$

using the definition of $\mathcal{J}^{\mathrm{GF}}$ and Lemma 3.8 for the third step. □

Lemma 4.10. *The redundancy criterion for* GH *is a redundancy criterion in the sense of the framework.*

Proof. We must prove conditions (R1) to (R4) defined by Waldmann et al.

(R1) It suffices to show that $N \setminus \mathit{GHRed}_{\mathrm{C}}(N) \models N$ for $N \subseteq \mathcal{C}_{\mathrm{GH}}$. Let $\mathcal{I}$ be a model of $N \setminus \mathit{GHRed}_{\mathrm{C}}(N)$. In the extensional variants, let $\mathcal{I}$ be extensional. Then by Lemma 4.9, there exists a model $\mathcal{I}^{\mathrm{GH}}$ of $\mathcal{F}(N \setminus \mathit{GHRed}_{\mathrm{C}}(N)) = \mathcal{F}(N) \setminus \mathit{GHRed}_{\mathrm{C}}(\mathcal{F}(N))$. By Lemma 5.2 of Bachmair and Ganzinger, this is also a model of $\mathcal{F}(N)$. By Lemma 4.9, it follows that $\mathcal{I} \models N$.

(R2) We must show that $N \subseteq N'$ implies that $\mathit{GHRed}_{\mathrm{C}}(N) \subseteq \mathit{GHRed}_{\mathrm{C}}(N')$ and $\mathit{GHRed}_{\mathrm{I}}(N) \subseteq \mathit{GHRed}_{\mathrm{I}}(N')$. By Lemma 5.6(i) of Bachmair and Ganzinger, this holds on GF. For clauses and all inferences except GArgCong and GExt, this implies that it holds on GH as well because $\mathcal{F}$ is a redundancy-preserving bijection between $\mathcal{C}_{\mathrm{GH}}$ and $\mathcal{C}_{\mathrm{GF}}$ and between these inferences. For GArgCong and GExt inferences, it holds because it holds on clauses.

(R3) We must show that if $N' \subseteq \mathit{GHRed}_{\mathrm{C}}(N)$, then $\mathit{GHRed}_{\mathrm{C}}(N) \subseteq \mathit{GHRed}_{\mathrm{C}}(N \setminus N')$ and $\mathit{GHRed}_{\mathrm{I}}(N) \subseteq \mathit{GHRed}_{\mathrm{I}}(N \setminus N')$. The proof is analogous to (R2), with Lemma 5.6(ii) of Bachmair and Ganzinger instead of Lemma 5.6(i).

(R4) We must show that for all inferences with $\mathit{concl}(\iota) \in N$, we have $\iota \in \mathit{GHRed}_{\mathrm{I}}(N)$. Since $\mathit{concl}(\iota) \prec \mathit{mprem}(\iota)$ for all $\iota \in \mathit{GFInf}$, this holds on GF. For all inferences except GArgCong and GExt, since $\mathcal{F}$ is a bijection preserving redundancy, it follows that it also holds also on GH. For GArgCong and GExt inferences, it holds by definition. □

Lemma 4.11. *Given a selection function sel on $\mathcal{C}_{\mathrm{H}}$, the grounding functions $\mathcal{G}^{\mathit{gsel}}$ for $\mathit{gsel} \in \mathcal{G}(\mathit{sel})$ are grounding functions in the sense of the framework.*

Proof. Clearly, $C = \bot$ if and only if $\mathcal{G}(C) = \bot$, proving (G1) and (G2). For (G3), we have to show for all non-PosExt inferences $\iota \in \mathit{HInf}^{\mathit{sel}}$ that $\mathcal{G}^{\mathit{gsel}}(\iota) \subseteq \mathit{GHRed}_{\mathrm{I}}^{\mathit{gsel}}(\mathcal{G}(\mathit{concl}(\iota)))$. Let $\iota \in \mathit{HInf}^{\mathit{sel}}$ and $\iota' \in \mathcal{G}^{\mathit{gsel}}(\iota)$. By the definition of $\mathcal{G}^{\mathit{gsel}}$, there exists a grounding substitution θ such that $\mathit{prems}(\iota') = \mathit{prems}(\iota)\theta$ and $\mathit{concl}(\iota') = \mathit{preconcl}(\iota)\theta$. We want to show that $\iota' \in \mathit{GHRed}_{\mathrm{I}}^{\mathit{gsel}}(\mathcal{G}(\mathit{concl}(\iota)))$.

If ι' is not an GArgCong or GExt inference, by the definition of inference redundancy, it suffices to show that $\{D \in \mathcal{F}(\mathcal{G}(\mathit{concl}(\iota))) \mid D \prec \mathit{mprem}(\mathcal{F}(\iota'))\} \models \mathit{concl}(\mathcal{F}(\iota'))$. We define a substitution θ' that extends θ to all variables in $\mathit{concl}(\iota)$. Due to purification, the clause $\mathit{concl}(\iota)$ may contain variables not present in $\mathit{preconcl}(\iota)$. For each such variable x', let x be the variable in $\mathit{preconcl}(\iota)$ that x' stems from and define $x'\theta' = x\theta$. Then the clause $\mathcal{F}(\mathit{concl}(\iota)\theta')$ differs from the clause $\mathcal{F}(\mathit{concl}(\iota')) = \mathcal{F}(\mathit{preconcl}(\iota)\theta')$ only in some additional grounded purification literals, which all have the form $t \not\approx t$ and are thus trivially false in any interpretation. Hence, $\mathcal{F}(\mathit{concl}(\iota)\theta') \models \mathcal{F}(\mathit{concl}(\iota'))$. Since one of the variables of a purification literal must appear applied in the clause, for each grounded purification literal $t \not\approx t$ the term t must be smaller than the maximal term of the clause $\mathcal{F}(\mathit{concl}(\iota'))$.

If no literals are selected in $\mathit{mprem}(\mathcal{F}(\iota'))$, inspection of the rules in *GFInf* shows that $\mathcal{F}(\mathit{concl}(\iota)\theta') \prec \mathit{mprem}(\mathcal{F}(\iota'))$. Otherwise, ι' can only be an EqRes inference or a Sup inference into a negative literal. If it is an EqRes inference, due to the selection restrictions, the substitution σ used in ι is the identity for all variables of functional type. Therefore, applying σ cannot trigger any purification and hence $\mathcal{F}(\mathit{concl}(\iota)\theta') = \mathcal{F}(\mathit{preconcl}(\iota)\theta') \prec \mathit{mprem}(\mathcal{F}(\iota'))$. If ι' is a Sup inference into a negative literal, due to the selection restrictions, the substitution $\sigma = \mathrm{mgu}(t, u)$ used in ι is the identity for all variables of functional type that stem from the main premise. Therefore the variables from the main premise C need not be purified. The variables from the side premise D might need to be purified, yielding purification literals of the form $x \not\approx y$ where $x\theta' = y\theta'$. Then x or y must appear applied in D and hence $x\theta'$ is smaller than $t\theta'$. Again, it follows that $\mathcal{F}(\mathit{concl}(\iota)\theta') \prec \mathit{mprem}(\mathcal{F}(\iota'))$.

This proves $\{D \in \mathcal{F}(\mathcal{G}(\mathit{concl}(\iota))) \mid D \prec \mathit{mprem}(\mathcal{F}(\iota'))\} \models \mathit{concl}(\mathcal{F}(\iota'))$.

In the nonpurifying variants, if ι' is an GArgCong or GExt inference, it suffices to show that $\mathit{concl}(\iota') \in \mathcal{G}(\mathit{concl}(\iota))$. This holds because $\mathit{concl}(\iota') = \mathit{preconcl}(\iota)\theta = \mathit{concl}(\iota)\theta$. In the purifying variants, if ι' is an GArgCong or GExt inference, we must show that

$\mathcal{F}(\mathcal{G}(\mathit{concl}(\iota))) \models \mathcal{F}(\mathit{concl}(\iota'))$. Defining θ' as above, we have $\mathcal{F}(\mathit{concl}(\iota)\theta') \models \mathcal{F}(\mathit{concl}(\iota'))$, as desired. □

Let *sel* be a selection function on $\mathcal{C}_{\mathrm{H}}$ fulfilling the selection restrictions introduced in Section 3.1. Let $N \subseteq \mathcal{C}_{\mathrm{H}}$ be a clause set saturated w.r.t. $\mathit{HInf}^{\mathit{sel}}$ and $\mathit{HRed}_{\mathrm{I}}^{\mathit{sel}}$. For the lifting mechanism of the saturation framework to apply, we need to show that there exists a selection function $\mathit{gsel} \in \mathcal{G}(\mathit{sel})$ such that all inferences $\iota \in \mathit{GHInf}^{\mathit{gsel}}$ with $\mathit{prems}(\iota) \in \mathcal{G}(N)$ are liftable or redundant. Here, by *liftable*, we mean that ι is a $\mathcal{G}^{\mathit{gsel}}$-ground instance of a $\mathit{HInf}^{\mathit{sel}}$-inference from N; by *redundant*, we mean that $\iota \in \mathit{GHRed}_{\mathrm{I}}^{\mathit{gsel}}(\mathcal{G}(N))$.

To choose the right selection function $\mathit{gsel} \in \mathcal{G}(\mathit{sel})$, we observe that each ground clause $C \in \mathcal{G}(N)$ must have at least one corresponding clause $D \in N$ such that C is a ground instance of D. We choose one of them for each $C \in \mathcal{G}(N)$, which we denote by $\mathcal{G}^{-1}(C)$. Then let *gsel* select those literals in C that correspond to the literals selected by *sel* in $\mathcal{G}^{-1}(C)$. Given this selection function *gsel*, we can show that all inferences from $\mathcal{G}(N)$ are liftable or redundant.

All non-Sup inferences in *GHInf* are liftable (Lemma 4.13). For Sup, some inferences are liftable (Lemma 4.14) and some are redundant (Lemma 4.15). As in standard superposition, Sup inferences into positions below variables are redundant. The variable condition of each of the four calculi is designed to cover the nonredundant Sup inferences into positions of variables or applied variables, which makes these inferences liftable.

Lemma 4.12. *Let σ be the most general unifier of s and s'. Let θ be an arbitrary unifier of s and s'. Then $\sigma\theta = \theta$.*

Proof. Like in first-order logic, we can assume that σ is idempotent without loss of generality [34, Corollary 7.2.11]. Since σ is most general, there exists a substitution ρ such that $\sigma\rho = \theta$. Therefore, by idempotence, $\sigma\theta = \sigma\sigma\rho = \sigma\rho = \theta$. □

Lemma 4.13 (Lifting of EqRes, EqFact, GArgCong, and GExt)**.** *All* EqRes, EqFact, GArgCong, *and* GExt *inferences are liftable.*

Proof. EqRes: Let $\iota \in \mathit{GHInf}^{\mathit{gsel}}$ be an EqRes inference with $\mathit{prems}(\iota) \in \mathcal{G}(N)$. Then ι is of the form

$$\frac{C\theta \;=\; C'\theta \vee s\theta \not\approx s'\theta}{C'\theta}\,\textsc{EqRes}$$

where $\mathcal{G}^{-1}(C\theta) = C = C' \vee s \not\approx s'$ and the literal $s\theta \not\approx s'\theta$ is eligible w.r.t. *gsel*. Let $\sigma = \mathrm{mgu}(s, s')$. It follows that $s \not\approx s'$ is eligible in C w.r.t. σ and *sel*. Moreover, $s\theta$ and $s'\theta$ are unifiable and ground, and therefore $s\theta = s'\theta$. Thus, the following inference $\iota' \in \mathit{HInf}^{\mathit{sel}}$ is applicable:

$$\frac{C' \vee s \not\approx s'}{\mathit{pure}(C'\sigma)}\,\textsc{EqRes}$$

(where *pure* is the identity in nonpurifying variants). By Lemma 4.12, we have $C'\sigma\theta = C'\theta$. Therefore, ι is the θ-ground instance of ι' and is therefore liftable.

EqFact: Analogously, if $\iota \in \mathit{GHInf}^{\mathit{gsel}}$ is an EqFact inference with $\mathit{prems}(\iota) \in \mathcal{G}(N)$, then ι is of the form

$$\frac{C\theta \;=\; C'\theta \vee s'\theta \approx t'\theta \vee s\theta \approx t\theta}{C'\theta \vee t\theta \not\approx t'\theta \vee s\theta \approx t'\theta}\,\textsc{EqFact}$$

where $\mathcal{G}^{-1}(C\theta) = C = C' \vee s' \approx t' \vee s \approx t$, the literal $s\theta \approx t\theta$ is eligible in C w.r.t. *gsel*, and $s\theta \not\approx t\theta$. Let $\sigma = \mathrm{mgu}(s, s')$. Hence, $s \approx t$ is eligible in C w.r.t. σ and *sel*. We have $s \not\approx t$. Moreover, $s\theta$ and $s'\theta$ are unifiable and ground. Hence, $s\theta = s'\theta$. Thus, the following inference $\iota' \in \mathit{HInf}^{\mathit{sel}}$ is applicable:

$$\frac{C' \vee s' \approx t' \vee s \approx t}{\mathit{pure}((C' \vee t \not\approx t' \vee s \approx t')\sigma)}\,\textsc{EqFact}$$

By Lemma 4.12, we have $\mathit{preconcl}(\iota')\theta = \mathit{concl}(\iota)$. Hence, ι is the θ-ground instance of ι' and is therefore liftable.

GArgCong: Let $\iota \in \mathit{GHInf}^{\mathit{gsel}}$ be a GArgCong inference with $\mathit{prems}(\iota) \in \mathcal{G}(N)$. Then ι is of the form

$$\frac{C\theta \;=\; C'\theta \vee s\theta \approx s'\theta}{C'\theta \vee s\theta\, \bar{u}_n \approx s'\theta\, \bar{u}_n}\,\textsc{GArgCong}$$

where $\mathcal{G}^{-1}(C\theta) = C = C' \vee s \approx s'$, the literal $s\theta \approx s'\theta$ is strictly eligible w.r.t. *gsel*, and $s\theta$ and $s'\theta$ are of functional type. It follows that s and s' have either a functional or a polymorphic type. Let σ be the most general substitution such that $s\sigma$ and $s'\sigma$ take n arguments. Then $s \not\approx s'$ is eligible in C w.r.t. σ and *sel*. Hence the following inference $\iota' \in \mathit{HInf}^{\mathit{sel}}$ is applicable:

$$\frac{C' \vee s \approx s'}{\mathit{pure}((C' \vee s\, \bar{x}_n \approx s'\, \bar{x}_n)\sigma)}\,\textsc{ArgCong}$$

Then ι is a ground instance of ι' and is therefore liftable.

GExt: The conclusion of a GExt inference in *GHInf* is by definition a ground instance of the conclusion of the Ext inference in *HInf* before purification. Hence, the GExt inference is a ground instance of the Ext inference. Therefore it is liftable. □

Lemma 4.14 (Lifting of Sup)**.** *Let $\iota \in \mathit{GHInf}^{\mathit{gsel}}$ be a* Sup *inference*

$$\frac{D'\theta \vee t\theta \approx t'\theta \quad C'\theta \vee [\neg]\, s\theta\langle t\theta\rangle_p \approx s'\theta}{D'\theta \vee C'\theta \vee [\neg]\, s\theta\langle t'\theta\rangle_p \approx s'\theta}\,\textsc{Sup}$$

where $\mathcal{G}^{-1}(D\theta) = D = D' \vee t \approx t'$ and $\mathcal{G}^{-1}(C\theta) = C = C' \vee [\neg]\, s \approx s'$. Suppose that the position p exists as a green subterm in s. Let u be the green subterm of s at that position and $\sigma = \mathrm{mgu}(t, u)$ (which exists since θ is a unifier). If the variable condition holds for C, t, t', u, and σ, then ι is liftable.

Proof. The inference conditions of ι can be lifted to D and C. That $t\theta \approx t'\theta$ is strictly eligible in $D\theta$ w.r.t. *gsel* implies that $t \approx t'$ is strictly eligible in D w.r.t. σ and *sel*. If $[\neg]\, s\theta \approx s'\theta$ is (strictly) eligible in $C\theta$ w.r.t. *gsel*, then $[\neg]\, s \approx s'$ is (strictly) eligible in C w.r.t. σ and *sel*. Moreover, $D\theta \not\succeq C\theta$ implies $D \not\succeq C$, $t\theta \not\prec t'\theta$ implies $t \not\prec t'$, and $s\theta \not\prec s'\theta$ implies $s \not\prec s'$.

By assumption, p is a position of s and the variable condition holds. Thus, the following inference $\iota' \in \mathit{HInf}^{\mathit{sel}}$ is applicable:

$$\frac{D' \vee t \approx t' \quad C' \vee [\neg]\, s\langle u\rangle_p \approx s'}{\mathit{pure}((D' \vee C' \vee [\neg]\, s\langle t'\rangle_p \approx s')\sigma)}\,\textsc{Sup}$$

By Lemma 4.12, we have $(\mathit{preconcl}(\iota'))\theta = \mathit{concl}(\iota)$. Hence, ι is the θ-ground instance of ι' and is therefore liftable. □

The other Sup inferences might not be liftable, but they are redundant:

Lemma 4.15. *Let $\iota \in GHInf^{gsel}$ be a* Sup *inference*

$$\frac{D'\theta \vee t\theta \approx t'\theta \quad C'\theta \vee [\neg]\, s\theta\langle t\theta\rangle_p \approx s'\theta}{D'\theta \vee C'\theta \vee [\neg]\, s\theta\langle t'\theta\rangle_p \approx s'\theta}\,\textsc{Sup}$$

where $\mathcal{G}^{-1}(D\theta) = D = D' \vee t \approx t'$ and $\mathcal{G}^{-1}(C\theta) = C = C' \vee [\neg]\, s \approx s'$. Suppose that Lemma 4.14 does not apply. This could be either because the position p is below a variable in s or because the variable condition holds does not hold. Then $\iota \in GHRed_{\mathrm{I}}^{gsel}(\mathcal{G}(N))$.

Proof. By the definition of $GHRed_{\mathrm{I}}$, to show $\iota \in GHRed_{\mathrm{I}}^{gsel}(\mathcal{G}(N))$, it suffices to prove that $\{E \in \mathcal{F}(\mathcal{G}(N)) \mid E \prec \mathcal{F}(C\theta)\} \models \mathcal{F}(concl(\iota))$. Let $\mathcal{I}$ be a first-order model of all $E \in \mathcal{F}(\mathcal{G}(N))$ with $E \prec \mathcal{F}(C\theta)$. We must show that $\mathcal{I} \models \mathcal{F}(concl(\iota))$. If $\mathcal{I} \models \mathcal{F}(D'\theta)$, this is obvious. So we further assume that $\mathcal{I} \not\models \mathcal{F}(D'\theta)$. Since $D\theta \prec C\theta$ by the Sup inference conditions, it follows that $\mathcal{I} \models \mathcal{F}(t\theta \approx t'\theta)$. By congruence, it suffices to show $\mathcal{I} \models \mathcal{F}(C\theta)$. We proceed by a case distinction on the two possible reasons why Lemma 4.14 does not apply:

Case 1: The position p is below a variable in s. Then $t\theta$ is a proper green subterm of $x\theta$ and hence a green subterm of $x\theta\ \bar{w}$ for any arguments $\bar{w}$. Let v be the term that we obtain by replacing $t\theta$ by $t'\theta$ in $x\theta$ at the relevant position. It follows from our assumptions about $\mathcal{I}$ that $\mathcal{I} \models \mathcal{F}(t\theta \approx t'\theta)$, and by congruence, $\mathcal{I} \models \mathcal{F}(x\theta\ \bar{w} \approx v\ \bar{w})$ for any arguments $\bar{w}$. Hence, $\mathcal{I} \models \mathcal{F}(C\theta)$ if and only if $\mathcal{I} \models \mathcal{F}(C\{x \mapsto v\}\theta)$. By the inference conditions we have $t\theta \succ t'\theta$, which implies $\mathcal{F}(C\theta) \succ \mathcal{F}(C\{x \mapsto v\}\theta)$ by compatibility with green contexts. Therefore, we have $\mathcal{I} \models \mathcal{F}(C\{x \mapsto v\}\theta)$ and hence $\mathcal{I} \models \mathcal{F}(C\theta)$.

Case 2: The variable condition does not hold. In the extensional variants, it follows that u has a variable head and jells with $t \approx t'$. By Definition 3.1, this means that u, t, and t' have the following form: $u = x\ \bar{v}_n$ for some variable x and a tuple of terms $\bar{v}_n$ of length $n \geq 0$; $t = \tilde{t}\ \bar{x}_n$ and $t' = \tilde{t}'\ \bar{x}_n$, where $\bar{x}_n$ are variables that do not occur elsewhere in D.

For the intensional variants, we have $u \in \mathcal{V}$. Thus, u, t, and t' can be written in the same form as described above for the extensional variants, with $n = 0$.

Case 2.1 (Purifying calculi): First, we assume that x occurs only with arguments $\bar{v}_n$ in C. For the intensional variant, this must be the case because $n = 0$ and hence x can only occur without arguments by the definition of *pure* and the literal selection restriction. Define a substitution θ' by $x\theta' = \tilde{t}'\theta$ and $y\theta' = y\theta$ for other variables y. Since $t\theta \succ t'\theta$ by the inference conditions, we have $C\theta \succ C\theta'$. Moreover, $C\theta' \in \mathcal{G}(N)$. Then $\mathcal{I} \models \mathcal{F}(C\theta)$ by congruence, because $\mathcal{I} \models \mathcal{F}(C\theta')$ and $\mathcal{I} \models \mathcal{F}(t\theta \approx t'\theta)$.

Now we assume that x occurs with arguments other than $\bar{v}_n$ in C. This can only happen in the extensional variant and by the selection restrictions, $[\neg]\, s\theta \approx s'\theta$ may not be selected in $C\theta$. Therefore, $s\theta$ is the maximal term in $C\theta$. Then $s \neq x$ and hence $\bar{v}_n \neq \varepsilon$ because otherwise $s\theta = x\theta$ would be smaller than the applied occurrence of $x\theta$ in $C\theta$.

Define a substitution θ'' such that $x\theta'' = \tilde{t}'\theta$, $y\theta'' = \tilde{t}'\theta$ for other variables y if $y\theta = s\theta$ and C contains the literal $x \not\approx y$, and $y\theta'' = y\theta$ otherwise.

We show that $C\theta \succ C\theta''$ by proving that no literal of $C\theta''$ is larger than the maximal literal $[\neg]\, s\theta \approx s'\theta$ of $C\theta$ and that $[\neg]\, s\theta \approx s'\theta$ appears more often in $C\theta$ than in $C\theta''$.

For a literal of the form $x \not\approx y$, we have $x\theta'' \prec s\theta$ and $y\theta'' \prec s\theta$. For literals that are not of this form, by the definition of *pure* in the extensional variant, x occurs always

with arguments $\bar{v}_n$. Hence these literals are equal or smaller in $C\theta''$ than in $C\theta$, because $x\theta''\ \bar{v}_n \prec x\theta\ \bar{v}_n$ and $y\theta'' \preceq y\theta$. Therefore, no literal of $C\theta''$ is larger than the maximal literal $[\neg]\, s\theta \approx s'\theta$ of $C\theta$. Moreover, these inequalities show that every occurrence of $[\neg]\, s\theta \approx s'\theta$ in $C\theta''$ corresponds to an occurrence of $[\neg]\, s\theta \approx s'\theta$ in $C\theta$ that corresponds to a literal in C without the variable x. Since at least one occurrence of $[\neg]\, s\theta \approx s'\theta$ in $C\theta$ corresponds to a literal in C containing x, $[\neg]\, s\theta \approx s'\theta$ appears more often in $C\theta$ than in $C\theta''$. This concludes the argument that $C\theta \succ C\theta''$. It follows that $\mathcal{I} \models \mathcal{F}(C\theta'')$.

We need to show that $\mathcal{I} \models \mathcal{F}(C\theta)$. There is a PosExt inference from D to $D' \vee \tilde{t} \approx \tilde{t}'$. This inference is in $HRed_{\mathrm{I}}(N)$ because N is saturated. Therefore, $D'\theta \vee \tilde{t}\theta \approx \tilde{t}'\theta$ is in $\mathcal{G}(N) \cup GHRed_{\mathrm{C}}(\mathcal{G}(N))$. It follows that $\mathcal{I} \models \mathcal{F}(D'\theta \vee \tilde{t}\theta \approx \tilde{t}'\theta)$ because this clause is smaller than $\mathcal{F}(D'\theta)$ and hence smaller than $\mathcal{F}(C\theta)$. Since $\mathcal{F}(D'\theta)$ is false in $\mathcal{I}$, we have $\mathcal{I} \models \mathcal{F}(\tilde{t}\theta \approx \tilde{t}'\theta)$.

For every literal of the form $x \not\approx y$, where $y\theta = s\theta$, the variable y can only occur without arguments in C because of the maximality of $s\theta$. We distinguish two cases. If for every literal of the form $x \not\approx y$ where $y\theta = s\theta$, we have $\mathcal{I} \models \mathcal{F}(y\theta'' \approx y\theta)$, then $\mathcal{I} \models \mathcal{F}(C\theta)$ by congruence. If for some literal of the form $x \not\approx y$ where $y\theta = s\theta$, we have $\mathcal{I} \models \mathcal{F}(y\theta'' \not\approx y\theta)$, then $\mathcal{I} \models \mathcal{F}(y\theta \not\approx x\theta)$ because $y\theta'' = \tilde{t}'\theta$, $\mathcal{I} \models \mathcal{F}(\tilde{t}'\theta \approx \tilde{t}\theta)$, and $\tilde{t}\theta = x\theta$. Hence a literal of $\mathcal{F}(C\theta)$ is true in $\mathcal{I}$ and therefore $\mathcal{I} \models C\theta$.

Case 2.2 (Nonpurifying calculi): Since the variable condition does not hold, we have $C\theta \succeq C''\theta$, where $C'' = C\{x \mapsto \tilde{t}'\}$. We cannot have $C\theta = C''\theta$ because $x\theta = \tilde{t}\theta \neq \tilde{t}'\theta$ and x occurs in C. Hence, we have $C\theta \succ C''\theta$.

By the definition of $\mathcal{I}$, $C\theta \succ C''\theta$ implies $\mathcal{I} \models \mathcal{F}(C''\theta)$. We will use equalities that are true in $\mathcal{I}$ to rewrite $\mathcal{F}(C\theta)$ into $\mathcal{F}(C''\theta)$, which implies $\mathcal{I} \models \mathcal{F}(C\theta)$ by congruence.

By saturation of N, for any well-typed m-tuple of fresh variables $\bar{z}$, we can use a PosExt with premise D (if $n > m$) or ArgCong inference with premise D (if $n < m$) or using D itself (if $n = m$) to show that $\mathcal{G}(D' \vee \tilde{t}\ \bar{z} \approx \tilde{t}'\ \bar{z}) \subseteq \mathcal{G}(N) \cup GHRed_{\mathrm{C}}(\mathcal{G}(N))$. Hence, $D'\theta \vee \tilde{t}\theta\ \bar{u} \approx \tilde{t}'\theta\ \bar{u}$ is in $\mathcal{G}(N) \cup GHRed_{\mathrm{C}}(\mathcal{G}(N))$ for any ground arguments $\bar{u}$.

We observe that whenever $\tilde{t}\theta\ \bar{u}$ and $\tilde{t}'\theta\ \bar{u}$ are smaller than the maximal term of $C\theta$ for some arguments $\bar{u}$, we have

$$\mathcal{I} \models \mathcal{F}(\tilde{t}\theta\ \bar{u}) \approx \mathcal{F}(\tilde{t}'\theta\ \bar{u}) \tag{$\dagger$}$$

To show this, we assume that $\tilde{t}\theta\ \bar{u}$ and $\tilde{t}'\theta\ \bar{u}$ are smaller than the maximal term of $C\theta$ and we distinguish two cases: If $t\theta$ is smaller than the maximal term of $C\theta$, all terms in $D'\theta$ are smaller than the maximal term of $C\theta$ and hence $D'\theta \vee \tilde{t}\theta\ \bar{u} \approx \tilde{t}'\theta\ \bar{u} \prec C\theta$. If, on the other hand, $t\theta$ is equal to the maximal term of $C\theta$, $\tilde{t}\theta\ \bar{u}$ and $\tilde{t}'\theta\ \bar{u}$ are smaller than $t\theta$. Hence $\tilde{t}\theta\ \bar{u} \approx \tilde{t}'\theta\ \bar{u} \prec t\theta \approx t'\theta$ and $D'\theta \vee \tilde{t}\theta\ \bar{u} \approx \tilde{t}'\theta\ \bar{u} \prec D\theta \prec C\theta$. In both cases, since $\mathcal{F}(D'\theta)$ is false in $\mathcal{I}$ by assumption, $\mathcal{I} \models \mathcal{F}(\tilde{t}\theta\ \bar{u}) \approx \mathcal{F}(\tilde{t}'\theta\ \bar{u})$.

We proceed by a case distinction on whether $s\theta$ appears in a selected or in a maximal literal of $C\theta$. In both cases we provide an algorithm that establishes the equivalence of $C\theta$ and $C''\theta$ via rewriting using ($\dagger$). This might seem trivial at first sight, but we can only use the equations ($\dagger$) if $\tilde{t}\theta\ \bar{u}$ and $\tilde{t}'\theta\ \bar{u}$ are smaller than the maximal term of $C\theta$. Moreover, $\bar{u}$ might itself contain positions where we want to rewrite, so the order of rewriting matters.

Case 2.2.1: $s\theta$ is the maximal side of a maximal literal of $C\theta$. Then, since $C\theta \succ C''\theta$, every term in $C\theta$ and in $C''\theta$ is smaller than or equal to $s\theta$. Let C_0 and $\tilde{C}_0$ be the clauses resulting from rewriting $\mathcal{F}(t\theta) \to \mathcal{F}(t'\theta)$ wherever possible in $\mathcal{F}(C\theta)$ and $\mathcal{F}(C''\theta)$, respectively. Since $\mathcal{F}(t\theta)$ is a subterm of $\mathcal{F}(s\theta)$, now every term in C_0 and $\tilde{C}_0$ is strictly smaller than $\mathcal{F}(s\theta)$.

We define $C_1, C_2, \ldots$ inductively as follows: Given C_i, choose a subterm of the form $\mathcal{F}(\tilde{t}\theta\ \bar{u})$ where $\tilde{t}\theta\ \bar{u} \succ \tilde{t}'\theta\ \bar{u}$ or of the form $\mathcal{F}(\tilde{t}'\theta\ \bar{u})$ where $\tilde{t}'\theta\ \bar{u} \succ \tilde{t}\theta\ \bar{u}$. Let C_{i+1} be the clause resulting from rewriting that subterm $\mathcal{F}(\tilde{t}\theta\ \bar{u})$ to $\mathcal{F}(\tilde{t}'\theta\ \bar{u})$ or that subterm $\mathcal{F}(\tilde{t}'\theta\ \bar{u})$ to $\mathcal{F}(\tilde{t}\theta\ \bar{u})$ in C_i, depending on which term was chosen. Analogously, we define $\tilde{C}_1, \tilde{C}_2, \ldots$ by applying the same algorithm to $\tilde{C}_0$. In both cases, the process terminates because $\succ$ is compatible with green contexts and well founded. Let C_* and $\tilde{C}_*$ be the respective final clauses.

The algorithm preserves the invariant that every term in C_i and $\tilde{C}_i$ is strictly smaller than $s\theta$. By congruence via (†), applied at every step of the algorithm, we know that C_* and $\mathcal{F}(C\theta)$ are equivalent in $\mathcal{I}$ and that $\tilde{C}_*$ and $\mathcal{F}(C''\theta)$ are equivalent in $\mathcal{I}$ as well.

We show that $C_* = \tilde{C}_*$. Assume that $C_* \neq \tilde{C}_*$. The algorithm preserves a second invariant, namely that $\mathcal{F}^{-1}(C_i)$ and $\mathcal{F}^{-1}(\tilde{C}_j)$ are equal except for positions where one contains $\tilde{t}\theta$ and the other one contains $\tilde{t}'\theta$. Consider a deepest position where $\mathcal{F}^{-1}(C_*)$ and $\mathcal{F}^{-1}(\tilde{C}_*)$ are different. The respective position in C_* and $\tilde{C}_*$ then contains $\mathcal{F}(\tilde{t}\theta\ \bar{u})$ and $\mathcal{F}(\tilde{t}'\theta\ \bar{u})$ or vice versa. The arguments $\bar{u}$ must be equal because we consider a deepest position. But then $\tilde{t}\theta\ \bar{u} \succ \tilde{t}'\theta\ \bar{u}$ or $\tilde{t}\theta\ \bar{u} \prec \tilde{t}'\theta\ \bar{u}$, which is impossible since the algorithm terminated in C_* and $\tilde{C}_*$. This shows that $C_* = \tilde{C}_*$. Hence $\mathcal{F}(C\theta)$ and $\mathcal{F}(C''\theta)$ are equivalent, which proves $\mathcal{I} \models \mathcal{F}(C\theta)$.

Case 2.2.2: $s\theta$ is the maximal side of a selected literal of $C\theta$. Then, by the selection restrictions, x cannot be the head of a maximal literal of C.

At every position where $x\ \bar{u}$ occurs in C with some (or no) arguments $\bar{u}$, we rewrite $(\tilde{t}\ \bar{u})\theta$ to $(\tilde{t}'\ \bar{u})\theta$ in $C\theta$ if $(\tilde{t}\ \bar{u})\theta \succ (\tilde{t}'\ \bar{u})\theta$. We start with the innermost occurrences of x, so that the order of the two terms at one step does not change by later rewriting. Analogously, at every position where $x\ \bar{u}$ occurs in C with some (or no) arguments $\bar{u}$, we rewrite $(\tilde{t}'\ \bar{u})\theta$ to $(\tilde{t}\ \bar{u})\theta$ in $C''\theta$ if $(\tilde{t}'\ \bar{u})\theta \succ (\tilde{t}\ \bar{u})\theta$, again starting with the innermost occurrences.

We never rewrite at the top level of the maximal term of $C\theta$ or $C''\theta$ because x cannot be the head of a maximal literal of C. The two resulting clauses are identical because $C\theta$ and $C''\theta$ only differ at positions where x occurs in C. The rewritten terms are all smaller than the maximal term of $C\theta$. With (†), this implies that $\mathcal{I} \models \mathcal{F}(C\theta)$ by congruence. □

With these properties of our inference systems in place, the saturation framework guarantees static and dynamic refutational completeness of $\mathit{HInf}^{\mathit{sel}}$ w.r.t. $\mathit{HRed}_{\mathrm{I}}^{\mathit{sel}}$. However, the framework gives us refutational completeness w.r.t. the Herbrand entailment $\models_{\mathcal{G}}$, defined as $N_1 \models_{\mathcal{G}} N_2$ if $\mathcal{G}(N_1) \models \mathcal{G}(N_2)$, whereas our semantics is Tarski entailment $\models$, defined as $N_1 \models N_2$ if any model of N_1 is a model of N_2. The following lemma repairs this mismatch:

Lemma 4.16. *For $N \subseteq \mathcal{C}_{\mathrm{H}}$, we have $N \models_{\mathcal{G}} \bot$ if and only if $N \models \bot$.*

Proof. By Lemma 3.9, any model of N is also a model of $\mathcal{G}(N)$—i.e., $N \not\models \bot$ implies $N \not\models_{\mathcal{G}} \bot$. For the other direction, we need to show that $N \not\models_{\mathcal{G}} \bot$ implies $N \not\models \bot$. Assume that $N \not\models_{\mathcal{G}} \bot$—i.e., $\mathcal{G}(N) \not\models \bot$. Then there is a model $\mathcal{I}$ of $\mathcal{G}(N)$. We must show that there exists a model of N—i.e., $N \not\models \bot$. Let $\mathcal{I}'$ be an interpretation derived from $\mathcal{I}$ by removing all universes that are not the denotation of a type in $\mathcal{T}\mathit{y}_{\mathrm{GH}}$ and removing all domain elements that are not the denotation of a term in $\mathcal{T}_{\mathrm{GH}}$, making $\mathcal{I}'$ term-generated. Clearly, in our clausal logic, this leaves the denotations of terms and the truth of clauses unchanged. Thus, $\mathcal{I}' \models \mathcal{G}(N)$. We will show that $\mathcal{I}' \models N$. Let $C \in N$. We want to show that C is true in $\mathcal{I}'$ for all valuations ξ. Fix a valuation ξ. By construction, for each variable x, there exists a ground term s_x such that $[\![s_x]\!]_{\mathcal{I}'} = \xi(x)$. Let ρ be the substitution that maps every free variable x in C to s_x. Then $\xi(x) = [\![s_x]\!]_{\mathcal{I}'} = [\![x\rho]\!]_{\mathcal{I}'}$ for all x. By treating the type variables of C in the same way, we can also achieve that $\xi(\alpha) = [\![\alpha\rho]\!]_{\mathcal{I}'}$ for all x. By Lemma 3.8,

$[\![t\rho]\!]_{\mathcal{I}'} = [\![t]\!]^{\xi}_{\mathcal{I}'}$ for all terms t and $[\![\tau\rho]\!]_{\mathcal{I}'} = [\![\tau]\!]^{\xi}_{\mathcal{I}'}$ for all types τ. Hence, $C\rho$ and C have the same truth value in $\mathcal{I}'$ for ξ. Since $\mathcal{I}' \models \mathcal{G}(N)$, $C\rho$ is true in $\mathcal{I}'$ and thus C is true in $\mathcal{I}'$ as well. □

Theorem 4.17 (Static refutational completeness)**.** *Let $N \subseteq \mathcal{C}_{\mathrm{H}}$ be a clause set saturated w.r.t. $\mathit{HInf}^{\mathit{sel}}$ and $\mathit{HRed}_{\mathrm{I}}^{\mathit{sel}}$. Then $N \models \bot$ if and only if $\bot \in N$.*

Proof. We apply Theorem 14 of Waldmann et al. We take H for **F**, GH for **G**, and $\mathcal{F}(\mathit{sel})$ for Q. It is easy to see that the entailment relation $\models$ on GH is a consequence relation in the sense of the framework. By Lemma 4.10 and 4.11, $(\mathit{GHRed}_{\mathrm{I}}^{\mathit{gsel}}, \mathit{GHRed}_{\mathrm{C}})$ is a redundancy criterion in the sense of the framework, and $\mathcal{G}^{\mathit{gsel}}$ are grounding functions in the sense of the framework, for all $\mathit{gsel} \in \mathcal{F}(\mathit{sel})$. The redundancy criterion $(\mathit{HRed}_{\mathrm{I}}^{\mathit{sel}}, \mathit{HRed}_{\mathrm{C}})$ matches exactly the intersected lifted redundancy criterion $\mathit{Red}^{\cap,\sqsupset}$ of Waldmann et al. By Theorem 4.8, $\mathit{GHInf}^{\mathit{gsel}}$ is statically refutationally complete for all $\mathit{gsel} \in \mathcal{F}(\mathit{sel})$. By Lemmas 4.13, 4.14, and 4.15, for every saturated $N \subseteq \mathcal{C}_{\mathrm{H}}$, there exists a selection function $\mathit{gsel} \in \mathcal{G}(\mathit{sel})$ such that all inferences $\iota \in \mathit{GHInf}^{\mathit{gsel}}$ with $\mathit{prems}(\iota) \in \mathcal{G}(N)$ either are $\mathcal{G}^{\mathit{gsel}}$-ground instances of $\mathit{HInf}^{\mathit{sel}}$-inferences from N or belong to $\mathit{GHRed}_{\mathrm{I}}^{\mathit{gsel}}(\mathcal{G}(N))$.

If $\sqsupset = \varnothing$, Theorem 14 of Waldmann et al. implies that if $N \subseteq \mathcal{C}_{\mathrm{H}}$ is a clause set saturated w.r.t. $\mathit{HInf}^{\mathit{sel}}$ and $\mathit{HRed}_{\mathrm{I}}^{\mathit{sel}}$, then $N \models_{\mathcal{G}} \bot$ if and only if $\bot \in N$. By Lemma 16 of Waldmann et al., this also holds if $\sqsupset \neq \varnothing$. By Lemma 4.16, this also holds for the Tarski entailment $\models$. That is, if $N \subseteq \mathcal{C}_{\mathrm{H}}$ is a clause set saturated w.r.t. $\mathit{HInf}^{\mathit{sel}}$ and $\mathit{HRed}_{\mathrm{I}}^{\mathit{sel}}$, then $N \models \bot$ if and only if $\bot \in N$. □

From static refutational completeness, we can easily derive dynamic refutational completeness. Let $(N_i)_i$ be a (finite or infinite) sequence over sets of clauses from $\mathcal{C}_{\mathrm{H}}$. Such a sequence is called a *derivation* if $N_i \setminus N_{i+1} \subseteq \mathit{HRed}_{\mathrm{C}}(N_{i+1})$ for all i. It is called *fair* if all HInf-inferences from clauses in $\bigcup_i \bigcap_{j \geq i} N_j$ are contained in $\bigcup_i \mathit{HRed}_{\mathrm{I}}(N_i)$.

Theorem 4.18 (Dynamic refutational completeness)**.** *For every fair derivation $(N_i)_i$ such that $N_0 \models \{\bot\}$, we have $\bot \in N_i$ for some i.*

Proof. By Theorem 17 of Waldmann et al., this follows from Theorem 4.17 and Lemma 4.16. □

5. Implementation

Zipperposition [28, 29] is an open source superposition-based theorem prover written in OCaml.[1] It was initially designed for polymorphic first-order logic with equality, as embodied by TPTP TF1 [19]. We will refer to this implementation as Zipperposition's first-order mode. Later, Cruanes extended the prover with a pragmatic higher-order mode with support for λ-abstractions and extensionality, without any completeness guarantees. We have now also implemented complete clausal λ-free higher-order modes based on the four calculi described in this article.

The pragmatic higher-order mode provided a convenient basis to implement our calculi. It employs higher-order term and type representations and orders. Its ad hoc calculus extensions are similar to our calculi. They include an ArgCong-like rule and a PosExt-like rule, and Sup inferences are performed only at green subterms. One of the bugs we found

[1] `https://github.com/sneeuwballen/zipperposition`

during our implementation work occurred because argument positions shift when applied variables are instantiated. We resolved this by numbering argument positions in terms from right to left.

To implement the λ-free higher-order mode, we restricted the unification algorithm to non-λ-abstractions. To satisfy the requirements on selection, we avoid selecting literals that contain higher-order variables. To comply with our redundancy notion, we disabled rewriting of non-green subterms. Finally, to improve term indexing of higher-order terms, we replaced the imperfect discrimination trees by fingerprint indices [55].

For the purifying calculi, we implemented purification as a simplification rule. This ensures that it is applied aggressively on all clauses, whether initial clauses from the problem or clauses produced during saturation, before any inferences are performed.

For the nonpurifying calculi, we added the possibility to perform Sup inferences at variable positions. This means that variables must be indexed as well. In addition, we modified the variable condition. However, it is in general impossible to decide whether there exists a ground substitution θ with $t\sigma\theta \succ t'\sigma\theta$ and $C\sigma\theta \prec C''\sigma\theta$. We overapproximate the condition as follows: (1) check whether x appears with different arguments in the clause C; (2) use an term-order-specific algorithm to determine whether there might exist a grounding substitution θ and terms $\bar{u}$ such that $t\sigma\theta \succ t'\sigma\theta$ and $t\sigma\theta\ \bar{u} \prec t'\sigma\theta\ \bar{u}$; and (3) check whether $C\sigma \not\succeq C''\sigma$. If these three conditions apply, we conclude that there might exist a ground substitution θ witnessing nonmonotonicity.

For the extensional calculi, we add axiom (Ext) to the clause set. To curb the explosion associated with extensionality, this axiom and all clauses derived from it are penalized by the clause selection heuristic. We also added the NegExt rule described in Section 3.2, which resembles Vampire's extensionality resolution rule [37].

The ArgCong rule can have infinitely many conclusions on polymorphic clauses. To capture this in the implementation, we store these infinite sequences of conclusions in the form of finite instructions of how to obtain the actual clauses. In addition to the usual active and passive set of the DISCOUNT loop architecture [3], we use a set of scheduled inferences that stores these instructions. We visit the scheduled inferences in this additional set and the clauses in the passive set fairly to achieve dynamic completeness of our prover architecture. Waldmann et al. [66, Example 34] and Bentkamp et al. [9, Section 6] describe this architecture in more detail.

Using Zipperposition, we can quantify the disadvantage of the applicative encoding on Example 3.7. A well-chosen KBO instance with argument coefficients allows Zipperposition to derive $\bot$ in 4 iterations of the prover's main loop and 0.03 s. KBO or LPO with default settings needs 203 iterations and 0.4 s, whereas KBO or LPO on the applicatively encoded problem needs 203 iterations and more than 1 s due to the larger terms.

6. Evaluation

We evaluated Zipperposition's implementation of our four calculi on Sledgehammer-generated Isabelle/HOL benchmarks [23] and on benchmarks from the TPTP library [62, 63]. We compare our calculi with an applicative encoding mode, which performs the applicative encoding after the clausal normal form transformation and then proceeds with Zipperposition's first-order mode. The encoding makes all function symbols nullary and replaces all applications with a polymorphic binary `app` symbol.

Our experimental data is available online.[2] We used the development version of Zipperposition, revision `2031e216`.[3] Since the present work is only a stepping stone towards a prover for full higher-order logic, it would be misleading to compare this prototype with state-of-the-art higher-order provers that support a stronger logic. Many of the higher-order problems in the TPTP library are in fact satisfiable for our λ-free logic, even though they may be unsatisfiable for full higher-order logic and labeled as such in the TPTP.

We instantiated all calculus variants with three different term orders: LPO [20], KBO [6] (without argument coefficients), and EPO [8]. Among these, LPO is the only nonmonotonic order and therefore the most relevant option to evaluate our calculi, which are designed to cope with nonmonotonicity. KBO and EPO provide a yardstick to assess the cost of nonmonotonicity. However, when using a monotonic order, it may be more efficient to superpose at non-argument subterms directly instead of relying on the ArgCong rule.

The Sledgehammer benchmarks, corresponding to Isabelle's Judgment Day suite, were regenerated to target clausal λ-free higher-order logic. They comprise 2506 problems in total, divided in two groups based on the number of Isabelle facts (lemmas, definitions, etc.) selected for inclusion in each problem: either 16 facts (SH16) or 256 facts (SH256). The problems were generated by encoding λ-expressions as λ-lifted supercombinators [47].

From the TPTP library, we collected 708 first-order problems in TFF format and 717 higher-order problems in THF format, both groups containing both monomorphic and polymorphic problems. We excluded all problems that contain interpreted arithmetic symbols, the symbols `(@@+)`, `(@@-)`, `(@+)`, `(@-)`, `(&)`, or tuples, as well as the `SYN000` problems, which are only intended to test the parser, and problems whose clausal normal form takes longer than 15 s to compute or falls outside the λ-free fragment described in Section 2.

Figure 1 summarizes, for the intensional calculi, the number of solved satisfiable and unsatisfiable problems within 180 s, and the time taken to show unsatisfiability. Figure 2 presents the corresponding data for the extensional calculi. The average time is computed over the problems that all configurations for the respective benchmark set and term order found to be unsatisfiable within the time limit. For each combination of benchmark set and term order, the best result is highlighted in bold. The evaluation was carried out on StarExec Iowa [60] using Intel Xeon E5-2609 0 CPUs clocked at 2.40 GHz.

The experimental results on the TFF part of the TPTP library confirm that our calculi handle the vast majority of problems that are solvable in first-order mode gracefully. On first-order problems, the calculi are occasionally at variance with the first-order mode, due to the interaction of ArgCong with polymorphic types and due to the extensionality axiom (Ext). In contrast, the applicative encoding is comparatively inefficient on problems that are already first-order. For LPO, the success rate drops by about 15%, and the average time to show unsatisfiability triples.

The SH16 benchmarks consist mostly of small, mildly higher-order problems. The small number of axioms benefits the applicative encoding enough to outperform the purifying variants but not the nonpurifying ones. The SH256 benchmarks are also mildly higher-order but much larger. Such problems are underrepresented in the TPTP library. On these, our calculi clearly outperform the applicative encoding. This is hardly surprising given that the proving effort is dominated by first-order reasoning, which they can perform gracefully.

The THF benchmarks generally require more sophisticated higher-order reasoning than the Sledgehammer benchmarks. On the THF set, the results are less clear. The applicative

[2] `http://matryoshka.gforge.inria.fr/pubs/lfhosup_article_data/`

[3] `https://github.com/sneeuwballen/zipperposition/tree/2031e216c1941acd76187882a073e8f1e533`

		# sat			# unsat			⌀ time		
		LPO	KBO	EPO	LPO	KBO	EPO	LPO	KBO	EPO
SH16	applicative encoding	111	**189**	65	373	382	157	0.9	1.2	10.7
	nonpurifying calculus	**136**	165	**133**	**383**	**385**	**381**	**0.4**	**0.3**	**0.0**
	purifying calculus	82	98	82	363	363	355	1.3	2.0	**0.0**
SH256	applicative encoding	**1**	**1**	**1**	471	488	36	9.4	8.7	63.8
	nonpurifying calculus	**1**	**1**	**1**	**543**	**554**	**498**	**2.3**	**2.3**	**0.1**
	purifying calculus	**1**	**1**	**1**	523	528	484	2.6	3.4	0.5
TFF	first-order mode	0	0	0	**212**	**229**	**107**	**1.9**	**2.3**	**1.5**
	applicative encoding	0	0	0	180	205	21	7.0	10.0	4.6
	nonpurifying calculus	0	0	0	210	**229**	105	**1.9**	2.4	**1.5**
	purifying calculus	0	0	0	211	**229**	105	2.1	2.6	1.6
THF	applicative encoding	**127**	**115**	111	523	522	428	0.9	0.6	0.8
	nonpurifying calculus	111	114	**112**	**529**	**527**	**516**	**0.3**	**0.3**	**0.0**
	purifying calculus	108	109	108	528	526	514	**0.3**	0.5	**0.0**

FIGURE 1. Evaluation of the intensional calculi

		# sat			# unsat			⌀ time		
		LPO	KBO	EPO	LPO	KBO	EPO	LPO	KBO	EPO
SH16	applicative encoding	79	**152**	48	379	386	157	1.2	1.3	11.4
	nonpurifying calculus	**103**	131	**95**	**386**	**393**	**387**	**0.4**	**0.1**	**0.0**
	purifying calculus	32	57	32	367	365	363	2.0	1.7	**0.0**
SH256	applicative encoding	**1**	**1**	**1**	462	486	36	7.5	9.4	63.8
	nonpurifying calculus	**1**	**1**	**1**	**548**	**572**	**504**	**1.9**	**2.1**	**0.1**
	purifying calculus	**1**	**1**	**1**	512	529	482	2.2	5.0	**0.1**
TFF	first-order mode	0	0	0	**212**	**229**	**107**	**1.9**	**2.5**	**1.5**
	applicative encoding	0	0	0	178	202	21	7.9	11.7	4.7
	nonpurifying calculus	0	0	0	207	**229**	106	2.1	3.0	**1.5**
	purifying calculus	0	0	0	210	**229**	105	2.2	3.2	1.6
THF	applicative encoding	**108**	**109**	105	526	527	436	0.9	0.6	1.1
	nonpurifying calculus	106	108	**107**	**539**	**535**	**526**	**0.3**	**0.3**	**0.0**
	purifying calculus	96	97	96	530	529	519	**0.3**	0.6	**0.0**

FIGURE 2. Evaluation of the extensional calculi

encoding and our calculi are roughly neck-and-neck. The nonpurifying variants detect unsatisfiability slightly more frequently, whereas the applicative encoding tends to find more saturations. It seems that, due to the large amount of higher-order reasoning necessary to solve TPTP problems, the advantage of our calculi on the first-order parts of the derivation

is not a decisive factor on these benchmarks. Across all benchmarks, the nonpurifying calculi outperform their purifying relatives.

KBO tends to have a slight advantage over LPO on all benchmark sets. But the gap between KBO and LPO is not larger on the higher-order benchmarks than on TFF. Since LPO is monotonic on first-order terms but nonmonotonic on higher-order terms, whereas KBO is monotonic on both, this suggests that there is no substantial cost associated with nonmonotonicity. EPO generally performs worse than the other two orders, with the exception of the nonpurifying variant on SH16 benchmarks, where it is roughly neck-and-neck with RPO. This suggests that for small, mildly higher-order problems, EPO can be a viable RPO-like complement to KBO if one considers the effort to implement our calculi too high.

7. Discussion and Related Work

Our calculi join a long list of extensions and refinements of superposition. Among the most closely related is Peltier's [51] Isabelle/HOL formalization of the refutational completeness of a superposition calculus that operates on λ-free higher-order terms and that is parameterized by a monotonic term order. Extensions with polymorphism and induction, independently developed by Cruanes [28, 29] and Wand [67], contribute to increasing the power of automatic provers. Detection of inconsistencies in axioms, as suggested by Schulz et al. [57], is important for large axiomatizations.

Also of interest is Bofill and Rubio's [22] integration of nonmonotonic orders in ordered paramodulation, a precursor of superposition. Their work is a veritable tour de force, but it is also highly complicated and restricted to ordered paramodulation. Lack of compatibility with arguments being a mild form of nonmonotonicity, it seemed preferable to start with superposition, enrich it with an ArgCong rule, and tune the side conditions until we obtained a complete calculus.

Most complications can be avoided by using a monotonic order such as KBO without argument coefficients. However, coefficients can be useful to help achieve compatibility with β-reduction. For example, the term $\lambda x.\ x + x$ could be treated as a constant with a coefficient of 2 on its argument and a heavy weight to ensure $(\lambda x.\ x + x)\ y \succ y + y$. Although they do not use argument coefficients, the recently developed λ-superposition calculus by Bentkamp et al. [10] and combinatory superposition calculus by Bhayat and Reger [16] need a nonmonotonic order to cope with β-reduction. They are modeled after our extensional nonpurifying and intensional nonpurifying calculi, respectively.

Many researchers have proposed or used encodings of higher-order logic constructs into first-order logic, including Robinson [53], Kerber [42], Dougherty [32], Dowek et al. [33], Hurd [41], Meng and Paulson [47], Obermeyer [50], and Czajka [30]. Encodings of types, such as those by Bobot and Paskevich [21] and Blanchette et al. [17], are also crucial to obtain a sound encoding of higher-order logic. These ideas are implemented in proof assistant tools such as HOLyHammer and Sledgehammer [18].

In the term rewriting community, λ-free higher-order logic is known as applicative first-order logic. First-order rewrite techniques can be applied to this logic via the applicative encoding. However, there are similar drawbacks as in theorem proving to having `app` as the only nonnullary symbol. Hirokawa et al. [39] propose a technique that resembles our mapping $\mathcal{F}$ to avoid these drawbacks.

Another line of research has focused on the development of automated proof procedures for higher-order logic. Robinson's [52], Andrews's [1], and Huet's [40] pioneering work stands

out. Andrews [2] and Benzmüller and Miller [12] provide excellent surveys. The competitive higher-order automatic theorem provers include Leo-II [13] (based on RUE resolution), Satallax [25] (based on a tableau calculus and a SAT solver), agsyHOL [46] (based on a focused sequent calculus and a generic narrowing engine), Leo-III [59] (based on a pragmatic higher-order version of ordered paramodulation with no completeness guarantees), CVC4 and veriT [5] (both based on satisfiability modulo theories), and Vampire [15, 16] (based on superposition and SK-style combinators). The Isabelle proof assistant [49] (which includes a tableau reasoner and a rewriting engine) and its Sledgehammer subsystem also participate in the higher-order division of the CADE ATP System Competition [61].

Zipperposition is a convenient vehicle for experimenting and prototyping because it is easier to understand and modify than highly-optimized C or C++ provers. Our middle-term goal is to design higher-order superposition calculi, implement them in state-of-the-art provers such as E [56], SPASS [68], and Vampire [44], and integrate these in proof assistants to provide a high level of automation. With its stratified architecture, Otter-λ [7] is perhaps the closest to what we are aiming at, but it is limited to second-order logic and offers no completeness guarantees. As a first step, Vukmirović, Blanchette, Cruanes, and Schulz [65] have generalized E's data structures and algorithms to clausal λ-free higher-order logic, assuming a monotonic KBO [6].

8. Conclusion

We presented four superposition calculi for intensional and extensional clausal λ-free higher-order logic and proved them refutationally complete. The calculi nicely generalize standard superposition and are compatible with our λ-free higher-order LPO and KBO. Especially on large problems, our experiments confirm what one would naturally expect: that native support for partial application and applied variables outperforms the applicative encoding.

The new calculi reduce the gap between proof assistants based on higher-order logic and superposition provers. We can use them to reason about arbitrary higher-order problems by axiomatizing suitable combinators. But perhaps more importantly, they appear promising as a stepping stone towards complete, highly efficient automatic theorem provers for full higher-order logic. Indeed, the subsequent work by Bentkamp et al. [10], which introduces support for λ-expressions, and Bhayat and Reger [16], which works with SK-style combinators, is largely based on our nonpurifying calculi.

Acknowledgment. We are grateful to the maintainers of StarExec for letting us use their service. We want to thank Petar Vukmirović for his work on Zipperposition and for his advice. We thank Christoph Benzmüller, Sander Dahmen, Johannes Hölzl, Anders Schlichtkrull, Stephan Schulz, Alexander Steen, Geoff Sutcliffe, Andrei Voronkov, Daniel Wand, Christoph Weidenbach, and the participants in the 2017 Dagstuhl Seminar on Deduction beyond First-Order Logic for stimulating discussions. We also want to thank Christoph Benzmüller, Ahmed Bhayat, Alexander Steen, Mark Summerfield, Sophie Tourret, and the anonymous reviewers for suggesting several textual improvements. Bentkamp and Blanchette's research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka).

References

[1] Andrews, P.B.: Resolution in type theory. J. Symb. Log. 36(3), 414–432 (1971)
[2] Andrews, P.B.: Classical type theory. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. II, pp. 965–1007. Elsevier and MIT Press (2001)

[3] Avenhaus, J., Denzinger, J., Fuchs, M.: DISCOUNT: A system for distributed equational deduction. In: Hsiang, J. (ed.) RTA-95. LNCS, vol. 914, pp. 397–402. Springer (1995)
[4] Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. 4(3), 217–247 (1994)
[5] Barbosa, H., Reynolds, A., El Ouraoui, D., Tinelli, C., Barrett, C.: Extending SMT solvers to higher-order logic. In: Fontaine, P. (ed.) CADE-27. LNCS, vol. 11716, pp. 35–54. Springer (2019)
[6] Becker, H., Blanchette, J.C., Waldmann, U., Wand, D.: A transfinite Knuth–Bendix order for lambda-free higher-order terms. In: de Moura, L. (ed.) CADE-26. LNCS, vol. 10395, pp. 432–453. Springer (2017)
[7] Beeson, M.: Lambda logic. In: Basin, D.A., Rusinowitch, M. (eds.) IJCAR 2004. LNCS, vol. 3097, pp. 460–474. Springer (2004)
[8] Bentkamp, A.: Formalization of the embedding path order for lambda-free higher-order terms. Archive of Formal Proofs (2018), `http://isa-afp.org/entries/Lambda_Free_EPO.html`
[9] Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P., Waldmann, U.: Superposition with lambdas, to be submitted to a journal. `http://matryoshka.gforge.inria.fr/pubs/lamsup_article.pdf`
[10] Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P., Waldmann, U.: Superposition with lambdas. In: Fontaine, P. (ed.) CADE-27. LNCS, vol. 11716, pp. 55–73. Springer (2019)
[11] Bentkamp, A., Blanchette, J.C., Cruanes, S., Waldmann, U.: Superposition for lambda-free higher-order logic. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS, vol. 10900, pp. 28–46. Springer (2018)
[12] Benzmüller, C., Miller, D.: Automation of higher-order logic. In: Siekmann, J.H. (ed.) Computational Logic, Handbook of the History of Logic, vol. 9, pp. 215–254. Elsevier (2014)
[13] Benzmüller, C., Paulson, L.C., Theiss, F., Fietzke, A.: LEO-II—A cooperative automatic theorem prover for higher-order logic. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS, vol. 5195, pp. 162–170. Springer (2008)
[14] Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science, Springer (2004)
[15] Bhayat, A., Reger, G.: Restricted combinatory unification. In: Fontaine, P. (ed.) CADE-27. LNCS, vol. 11716, pp. 74–93. Springer (2019)
[16] Bhayat, A., Reger, G.: A combinator-based superposition calculus for higher-order logic. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS, Springer (2020)
[17] Blanchette, J.C., Böhme, S., Popescu, A., Smallbone, N.: Encoding monomorphic and polymorphic types. Log. Meth. Comput. Sci. 12(4:13), 1–52 (2016)
[18] Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.: Hammering towards QED. J. Formaliz. Reas. 9(1), 101–148 (2016)
[19] Blanchette, J.C., Paskevich, A.: TFF1: The TPTP typed first-order form with rank-1 polymorphism. In: Bonacina, M.P. (ed.) CADE-24. LNCS, vol. 7898, pp. 414–420. Springer (2013)
[20] Blanchette, J.C., Waldmann, U., Wand, D.: A lambda-free higher-order recursive path order. In: Esparza, J., Murawski, A.S. (eds.) FoSSaCS 2017. LNCS, vol. 10203, pp. 461–479. Springer (2017)
[21] Bobot, F., Paskevich, A.: Expressing polymorphic types in a many-sorted language. In: Tinelli, C., Sofronie-Stokkermans, V. (eds.) FroCoS 2011. LNCS, vol. 6989, pp. 87–102. Springer (2011)
[22] Bofill, M., Rubio, A.: Paramodulation with non-monotonic orderings and simplification. J. Autom. Reason. 50(1), 51–98 (2013)
[23] Böhme, S., Nipkow, T.: Sledgehammer: Judgement Day. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 107–121. Springer (2010)
[24] Brand, D.: Proving theorems with the modification method. SIAM J. Comput. 4, 412–430 (1975)
[25] Brown, C.E.: Satallax: An automatic higher-order prover. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 111–117. Springer (2012)
[26] Cervesato, I., Pfenning, F.: A linear spine calculus. J. Log. Comput. 13(5), 639–688 (2003)
[27] Church, A.: A formulation of the simple theory of types. J. Symb. Log. 5(2), 56–68 (1940)
[28] Cruanes, S.: Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond. Ph.D. thesis, École polytechnique (2015)
[29] Cruanes, S.: Superposition with structural induction. In: Dixon, C., Finger, M. (eds.) FroCoS 2017. LNCS, vol. 10483, pp. 172–188. Springer (2017)

[30] Czajka, Ł.: Improving automation in interactive theorem provers by efficient encoding of lambda-abstractions. In: Avigad, J., Chlipala, A. (eds.) CPP 2016. pp. 49–57. ACM (2016)
[31] Digricoli, V.J., Harrison, M.C.: Equality-based binary resolution. J. ACM 33(2), 253–289 (1986)
[32] Dougherty, D.J.: Higher-order unification via combinators. Theor. Comput. Sci. 114(2), 273–298 (1993)
[33] Dowek, G., Hardin, T., Kirchner, C.: Higher-order unification via explicit substitutions (extended abstract). In: LICS '95. pp. 366–374. IEEE Computer Society (1995)
[34] Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer-Verlag, 2nd edn. (1996)
[35] Fitting, M.: Types, Tableaus, and Gödel's God. Kluwer (2002)
[36] Gordon, M.J.C., Melham, T.F. (eds.): Introduction to HOL: A Theorem Proving Environment for Higher Order Logic. Cambridge University Press (1993)
[37] Gupta, A., Kovács, L., Kragl, B., Voronkov, A.: Extensional crisis and proving identity. In: Cassez, F., Raskin, J. (eds.) ATVA 2014. LNCS, vol. 8837, pp. 185–200. Springer (2014)
[38] Henkin, L.: Completeness in the theory of types. J. Symb. Log. 15(2), 81–91 (1950)
[39] Hirokawa, N., Middeldorp, A., Zankl, H.: Uncurrying for termination and complexity. J. Autom. Reasoning 50(3), 279–315 (2013)
[40] Huet, G.P.: A mechanization of type theory. In: Nilsson, N.J. (ed.) IJCAI-73. pp. 139–146. William Kaufmann (1973)
[41] Hurd, J.: First-order proof tactics in higher-order logic theorem provers. In: Archer, M., Di Vito, B., Muñoz, C. (eds.) Design and Application of Strategies/Tactics in Higher Order Logics. pp. 56–68. NASA Technical Reports (2003)
[42] Kerber, M.: How to prove higher order theorems in first order logic. In: Mylopoulos, J., Reiter, R. (eds.) IJCAI-91. pp. 137–142. Morgan Kaufmann (1991)
[43] Kop, C.: Higher Order Termination: Automatable Techniques for Proving Termination of Higher-Order Term Rewriting Systems. Ph.D. thesis, Vrije Universiteit Amsterdam (2012)
[44] Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer (2013)
[45] Leivant, D.: Higher order logic. In: Gabbay, D.M., Hogger, C.J., Robinson, J.A., Siekmann, J.H. (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 2, Deduction Methodologies, pp. 229–322. Oxford University Press (1994)
[46] Lindblad, F.: A focused sequent calculus for higher-order logic. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS, vol. 8562, pp. 61–75. Springer (2014)
[47] Meng, J., Paulson, L.C.: Translating higher-order clauses to first-order clauses. J. Autom. Reason. 40(1), 35–60 (2008)
[48] Miller, D.A.: A compact representation of proofs. Studia Logica 46(4), 347–370 (1987)
[49] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
[50] Obermeyer, F.H.: Automated Equational Reasoning in Nondeterministic λ-Calculi Modulo Theories $\mathcal{H}^*$. Ph.D. thesis, Carnegie Mellon University (2009)
[51] Peltier, N.: A variant of the superposition calculus. Archive of Formal Proofs (2016), `https://www.isa-afp.org/entries/SuperCalc.shtml`
[52] Robinson, J.: Mechanizing higher order logic. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. 4, pp. 151–170. Edinburgh University Press (1969)
[53] Robinson, J.: A note on mechanizing higher order logic. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence, vol. 5, pp. 121–135. Edinburgh University Press (1970)
[54] Schmidt-Schauß, M.: Unification in a combination of arbitrary disjoint equational theories. J. Symb. Comput. 8, 51–99 (1989)
[55] Schulz, S.: Fingerprint indexing for paramodulation and rewriting. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 477–483. Springer (2012)
[56] Schulz, S.: System description: E 1.8. In: McMillan, K.L., Middeldorp, A., Voronkov, A. (eds.) LPAR-19. LNCS, vol. 8312, pp. 735–743. Springer (2013)
[57] Schulz, S., Sutcliffe, G., Urban, J., Pease, A.: Detecting inconsistencies in large first-order knowledge bases. In: de Moura, L. (ed.) CADE-26. LNCS, vol. 10395, pp. 310–325. Springer (2017)
[58] Snyder, W., Lynch, C.: Goal directed strategies for paramodulation. In: Book, R.V. (ed.) RTA-91. LNCS, vol. 488, pp. 150–161. Springer (1991)

[59] Steen, A., Benzmüller, C.: The higher-order prover Leo-III. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS, vol. 10900, pp. 108–116. Springer (2018)
[60] Stump, A., Sutcliffe, G., Tinelli, C.: StarExec: A cross-community infrastructure for logic solving. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS, vol. 8562, pp. 367–373. Springer (2014)
[61] Sutcliffe, G.: The CADE-26 automated theorem proving system competition—CASC-26. AI Commun. 30(6), 419–432 (2017)
[62] Sutcliffe, G., Benzmüller, C., Brown, C.E., Theiss, F.: Progress in the development of automated theorem proving for higher-order logic. In: Schmidt, R.A. (ed.) CADE-22. LNCS, vol. 5663, pp. 116–130. Springer (2009)
[63] Sutcliffe, G., Schulz, S., Claessen, K., Baumgartner, P.: The TPTP typed first-order form with arithmetic. In: Bjørner, N., Voronkov, A. (eds.) LPAR-18. LNCS, vol. 7180, pp. 406–419. Springer (2012)
[64] Väänänen, J.: Second-order and higher-order logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, fall 2019 edn. (2019)
[65] Vukmirović, P., Blanchette, J.C., Cruanes, S., Schulz, S.: Extending a brainiac prover to lambda-free higher-order logic. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. pp. 192–210. LNCS, Springer (2019)
[66] Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS, Springer (2020)
[67] Wand, D.: Superposition: Types and Polymorphism. Ph.D. thesis, Universität des Saarlandes (2017)
[68] Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Schmidt, R.A. (ed.) CADE-22. LNCS, vol. 5663, pp. 140–145. Springer (2009)