Superposition for Full Higher-Order Logic (Technical Report)

Alexander Bentkamp[1], Jasmin Blanchette[1,2,3], Sophie Tourret[2,3], and Petar Vukmirović[1]

[1] Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
{a.bentkamp,j.c.blanchette,p.vukmirovic}@vu.nl
[2] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
sophie.tourret@inria.fr
[3] Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany

Abstract. We recently designed two calculi as stepping stones towards superposition for full higher-order logic: Boolean-free λ-superposition and superposition for first-order logic with interpreted Booleans. Stepping on these stones, we finally reach a sound and refutationally complete calculus for higher-order logic with polymorphism, extensionality, Hilbert choice, and Henkin semantics. In addition to the complexity of combining the calculus's two predecessors, new challenges arise from the interplay between λ-terms and Booleans. Our implementation in Zipperposition outperforms all other higher-order theorem provers and is on a par with an earlier, pragmatic prototype of Booleans in Zipperposition.

1 Introduction

Superposition is a leading calculus for first-order logic with equality. We have been wondering for some years whether it would be possible to gracefully generalize it to extensional higher-order logic and use it as the basis of a strong higher-order automatic theorem prover. Towards this goal, we have, together with colleagues, designed superposition-like calculi for three intermediate logics between first-order and higher-order logic. Now we are finally ready to assemble a superposition calculus for full higher-order logic. The filiation of our new calculus from Bachmair and Ganzinger's standard first-order superposition is as follows:

Standard superposition
Bachmair and Ganzinger [2] (Sup)

Superposition with $\longleftrightarrow$ and delayed CNF Ganzinger and Stuber [16] ($\longleftrightarrow$Sup)	Boolean-free λ-free superposition Bentkamp et al. [6] (λfSup)
Superposition with Booleans Nummelin et al. [27] (oSup)	Boolean-free λ-superposition Bentkamp et al. [7] (λSup)

Boolean λ-superposition
This report (oλSup)

Our goal was to devise an efficient calculus for higher-order logic. To achieve it, we pursued two objectives. First, the calculus should be refutationally complete. Second, the calculus should coincide as much as possible with its predecessors oSup and λSup on the respective fragments of higher-order logic (which in turn essentially coincide with Sup on first-order logic). Achieving these objectives is the main contribution of this paper. We made an effort to keep the calculus simple, but often the refutational completeness proof forced our hand to add conditions or special cases.

Like oSup, our calculus oλSup operates on clauses that can contain Boolean subterms, and it interleaves clausification with other inferences. Like λSup, oλSup eagerly $\beta\eta$-normalizes terms, employs full higher-order unification, and relies on a fluid subterm superposition rule (FLUIDSUP) to simulate superposition inferences below applied variables—i.e., terms of the form $y\,t_1 \ldots t_n$ for $n \geq 1$.

In addition to the issues discussed previously and the complexity of combining the two approaches, we encountered the following main challenges.

First, because oSup contains several superposition-like inference rules for Boolean subterms, our completeness proof requires dedicated *fluid Boolean subterm hoisting rules* (FLUIDBOOLHOIST, FLUIDLOOBHOIST), which simulate Boolean inferences below applied variables, in addition to FLUIDSUP, which simulates superposition inferences.

Second, due to restrictions related to the term order that parameterizes superposition, it is difficult to handle variables bound by unclausified quantifiers if these variables occur applied or in arguments of applied variables. We solve the issue by replacing such quantified terms $\forall y.\, t$ by equivalent terms $(\lambda y.\, t) \approx (\lambda y.\, \top)$ in a preprocessing step. We leave all other $\forall$-quantifiers intact so that we can process them more efficiently.

Third, like other higher-order calculi that support Booleans, our calculus must include some form of primitive substitution [1, 8, 18, 30]. For example, given the clauses $\mathsf{a} \not\approx \mathsf{b}$ and $z\,\mathsf{a} \approx \bot \vee z\,\mathsf{b} \approx \top$, it is crucial to find the substitution $\{z \mapsto \lambda v.\, v \approx \mathsf{a}\}$, which does not arise through unification. Primitive substitution accomplishes this by blindly substituting logical connectives and quantifiers; here it would apply $\{z \mapsto \lambda v.\, y\,v \approx y'\,v\}$ to the second clause, where y and y' are fresh variables. In the context of superposition, this is problematic because the instantiated clause is subsumed by the original clause and could be discarded. Our solution is to immediately clausify the introduced logical symbol, yielding a clause that is no longer subsumed.

We implemented our calculus in the Zipperposition prover and evaluated it on TPTP and Sledgehammer benchmarks. The new Zipperposition outperforms all other higher-order provers and is on a par with an ad hoc implementation of Booleans in the same prover by Vukmirović and Nummelin [34].

2 Logic

Our logic is higher-order logic (simple type theory) with rank-1 polymorphism, Hilbert choice, Henkin semantics, and functional and Boolean extensionality. It closely resembles Gordon and Melham's HOL [17] and the TPTP TH1 standard [22].

Although standard semantics is commonly considered the foundation of the HOL systems, also Henkin semantics is compatible with the notion of provability employed by the HOL systems. By admitting nonstandard models, Henkin semantics is not subject to Gödel's first incompleteness theorem, allowing us to claim refutational completeness of our calculus.

On top of the standard higher-order terms, we install a clausal structure that allows us to formulate calculus rules in the style of first-order superposition. It does not restrict the flexibility of the logic because an arbitrary term t of Boolean type can be written as the clause $t \approx \top$.

Syntax. We use the notation $\bar{a}_n$ or $\bar{a}$ to stand for the tuple $(a_1,\ldots,a_n)$ or product $a_1 \times \cdots \times a_n$, where $n \geq 0$. We abuse notation by applying an operation on a tuple when it must be applied elementwise; thus, $f(\bar{a}_n)$ can stand for $(f(a_1),\ldots,f(a_n))$, $f(a_1)\times\cdots\times f(a_n)$, or $f(a_1,\ldots,a_n)$, depending on the operation f.

As a basis for our logic's types, we fix an infinite set $\mathcal{V}_{\mathsf{ty}}$ of type variables. A set Σ_{ty} of type constructors with associated arities is a *type signature* if it contains at least one nullary Boolean type constructor o and a binary function type constructor $\rightarrow$. A *type*, usually denoted by τ or υ, is inductively defined to either be a type variable $\alpha \in \mathcal{V}_{\mathsf{ty}}$ or have the form $\kappa(\bar{\tau}_n)$ for an n-ary type constructor $\kappa \in \Sigma_{\mathsf{ty}}$ and types $\bar{\tau}_n$. We write κ for $\kappa()$ and $\tau \rightarrow \upsilon$ for $\rightarrow(\tau,\upsilon)$. A *type declaration* is an expression $\Pi\bar{\alpha}_m.\,\tau$ (or simply τ if $m = 0$), where all type variables occurring in τ belong to $\bar{\alpha}_m$.

To define our logic's terms, we fix a type signature Σ_{ty} and a set $\mathcal{V}$ of term variables with associated types, written as $x : \tau$ or x. We require that there are infinitely many variables for each type.

A *term signature* is a set Σ of (function) symbols, usually denoted by a, b, c, f, g, h, each associated with a type declaration, written as $\mathsf{f} : \Pi\bar{\alpha}_m.\,\tau$. We require the presence of the logical symbols $\boldsymbol{\top},\boldsymbol{\bot} : o$; $\boldsymbol{\neg} : o \rightarrow o$; $\boldsymbol{\wedge},\boldsymbol{\vee},\boldsymbol{\rightarrow} : o \rightarrow o \rightarrow o$; $\boldsymbol{\forall},\boldsymbol{\exists} : \Pi\alpha.\,(\alpha \rightarrow o) \rightarrow o$; and $\boldsymbol{\approx},\boldsymbol{\not\approx} : \Pi\alpha.\,\alpha \rightarrow \alpha \rightarrow o$. Moreover, we require the presence of the Hilbert choice operator $\varepsilon : \Pi\alpha.\,(\alpha \rightarrow o) \rightarrow \alpha$. The logical symbols are printed in bold to distinguish them from the notation used for clauses below. Although ε is interpreted in our semantics, we do not consider it to be a logical symbol. The reason is that our calculus will enforce the semantics of ε by an axiom, whereas the semantics of the logical symbols will be enforced by inference rules. In the following, we also fix a term signature Σ. A type signature and a term signature form a *signature*.

We will define terms in three layers of abstraction: raw λ-terms, λ-terms, and terms; where λ-terms will be α-equivalence classes of raw λ-terms and terms will be $\beta\eta$-equivalence classes of λ-terms.

We write $t : \tau$ for a raw lambda term t of type τ. The set of *raw λ-terms* and their associated types is defined inductively as follows. Every $x : \tau \in \mathcal{V}$ is a raw λ-term of type τ. If $\mathsf{f} : \Pi\bar{\alpha}_m.\,\tau \in \Sigma$ and $\bar{\upsilon}_m$ is a tuple of types, called *type arguments*, then $\mathsf{f}\langle\bar{\upsilon}_m\rangle$ (or simply f if $m = 0$) is a raw λ-term of type $\tau\{\bar{\alpha}_m \mapsto \bar{\upsilon}_m\}$. If $x : \tau$ and $t : \upsilon$, then the *λ-expression* $\lambda x.\,t$ is a raw λ-term of type $\tau \rightarrow \upsilon$. If $s : \tau \rightarrow \upsilon$ and $t : \tau$, then the *application* $s\,t$ is a raw λ-term of type υ. Using the spine notation [14], raw λ-terms can be decomposed in a unique way as a nonapplication *head* t applied to zero or more arguments: $t\,s_1 \ldots s_n$ or $t\,\bar{s}_n$ (abusing notation). For the symbols $\boldsymbol{\approx}$ and $\boldsymbol{\not\approx}$, we will typically use infix notation and omit the type argument.

A raw λ-term s is a *subterm* of a raw λ-term t, written $t = t[s]$, if $t = s$, if $t = (\lambda x.\,u[s])$, if $t = (u[s])\,v$, or if $t = u\,(v[s])$ for some raw λ-terms u and v. A *proper* subterm of a raw λ-term t is any subterm of t that is distinct from t itself. A variable occurrence is *free* in a raw λ-term if it is not bound by a λ-expression. A raw λ-term is *ground* if it is built without using type variables and contains no free term variables.

The α-renaming rule is defined as $(\lambda x.\, t) \longrightarrow_\alpha (\lambda y.\, t\{x \mapsto y\})$, where y does not occur free in t and is not captured by a λ-binder in t. Raw λ-terms form equivalence classes modulo α-renaming, called *λ-terms*. We lift the above notions on raw λ-terms to λ-terms.

A substitution ρ is a function from type variables to types and from term variables to λ-terms such that it maps all but finitely many variables to themselves. We require that it is type-correct—i.e., for each $x : \tau \in \mathcal{V}$, $x\rho$ is of type $\tau\rho$. The letters θ, ρ, σ are reserved for substitutions. Substitutions α-rename λ-terms to avoid capture; for example, $(\lambda x.\, y)\{y \mapsto x\} = (\lambda x'.\, x)$. The composition $\rho\sigma$ applies ρ first: $t\rho\sigma = (t\rho)\sigma$. The notation $\sigma[\bar{x}_n \mapsto \bar{s}_n]$ denotes the substitution that replaces each x_i by s_i and that otherwise coincides with σ.

The β- and η-reduction rules are specified on λ-terms as $(\lambda x.\, t)\, u \longrightarrow_\beta t\{x \mapsto u\}$ and $(\lambda x.\, t\, x) \longrightarrow_\eta t$. For β, bound variables in t are implicitly renamed to avoid capture; for η, the variable x may not occur free in t. The λ-terms form equivalence classes modulo $\beta\eta$-reduction, called *$\beta\eta$-equivalence classes* or simply *terms*.

We use the following nonstandard normal form: The *$\beta\eta\mathsf{Q}_\eta$-normal* form $t{\downarrow}_{\beta\eta\mathsf{Q}_\eta}$ of a λ-term t is obtained by applying $\longrightarrow_\beta$ and $\longrightarrow_\eta$ exhaustively and finally applying the following rewrite rule Q_η exhaustively:

$$\mathsf{Q}\langle\tau\rangle\, t \longrightarrow_{\mathsf{Q}_\eta} \mathsf{Q}\langle\tau\rangle\, (\lambda x.\, t\, x)$$

where t is not a λ-expression. Here and elsewhere, Q stands for either $\forall$ or $\exists$.

We lift all of the notions defined on λ-terms to terms:

Convention 1. When defining operations that need to analyze the structure of terms, we use the $\beta\eta\mathsf{Q}_\eta$-normal representative as the default representative of a $\beta\eta$-equivalence class.

Many authors prefer the η-long β-normal form [19,21,26], but in a polymorphic setting it has the drawback that instantiating a type variable with a functional type can lead to η-expansion.

A literal is an equation $s \approx t$ or a disequation $s \not\approx t$ of terms s and t. In both cases, the order of s and t is not fixed. We write $s \mathrel{\dot\approx} t$ for a literal that can be either an equation or a disequation. A clause $L_1 \vee \cdots \vee L_n$ is a finite multiset of literals L_j. The empty clause is written as $\bot$.

Our calculus does not allow nonequational literals. These must be encoded as $t \approx \mathbf{T}$ or $t \approx \mathbf{\bot}$. We even considered to exclude negative literals by encoding them as $(s \mathbin{\boldsymbol{\approx}} t) \approx \mathbf{\bot}$, following $\leftrightarrow$Sup [16]. However, this approach would make the conclusion of the equality factoring rule (EFACT) too large for our purposes. Regardless, the simplification machinery will allow us to reduce negative literals of the form $t \not\approx \mathbf{\bot}$ and $t \not\approx \mathbf{T}$ to $t \approx \mathbf{T}$ and $t \approx \mathbf{\bot}$, thereby eliminating redundant representations of literals.

A *complete set of unifiers* on a set X of variables for two terms s and t is a set U of unifiers of s and t such that for every unifier θ of s and t there exists a member $\sigma \in U$ and a substitution ρ such that $x\sigma\rho = x\theta$ for all $x \in X$. We let $\mathrm{CSU}_X(s,t)$ denote an arbitrary (preferably, minimal) complete set of unifiers on X for s and t. We assume that all $\sigma \in \mathrm{CSU}_X(s,t)$ are idempotent on X—i.e., $x\sigma\sigma = x\sigma$ for all $x \in X$. The set X will consist of the free variables of the clauses in which s and t occur and will be left implicit. To compute $\mathrm{CSU}(s,t)$, Huet-style preunification [18] is not sufficient, and we must resort to full unification procedures [20, 33].

Semantics. A *type interpretation* $\mathcal{I}_{\mathsf{ty}} = (\mathcal{U}, \mathcal{J}_{\mathsf{ty}})$ is defined as follows. The *universe* $\mathcal{U}$ is a collection of nonempty sets, called *domains*. We require that $\{0,1\} \in \mathcal{U}$. 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, such that $\mathcal{J}_{\mathsf{ty}}(o) = \{0,1\}$ and for all domains $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{U}$, the set $\mathcal{J}_{\mathsf{ty}}(\to)(\mathcal{D}_1, \mathcal{D}_2)$ is a subset of the function space from $\mathcal{D}_1$ to $\mathcal{D}_2$. The semantics is *standard* if $\mathcal{J}_{\mathsf{ty}}(\to)(\mathcal{D}_1, \mathcal{D}_2)$ is the entire function space for all $\mathcal{D}_1, \mathcal{D}_2$. A *type valuation* ξ is a function that maps every type variable to a domain. The *denotation* of a type for a type interpretation $\mathcal{I}_{\mathsf{ty}}$ and a type valuation ξ is recursively defined by $[\![\alpha]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} = \xi(\alpha)$ and $[\![\kappa(\bar{\tau})]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}} = \mathcal{J}_{\mathsf{ty}}(\kappa)([\![\bar{\tau}]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}})$.

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 domain tuple $\bar{\mathcal{D}}_m \in \mathcal{U}^m$ a value $\mathcal{J}(\mathsf{f}, \bar{\mathcal{D}}_m) \in [\![\tau]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$, where ξ is the type valuation that maps each α_i to $\mathcal{D}_i$. We require that

(I1) $\mathcal{J}(\top) = 1$
(I2) $\mathcal{J}(\bot) = 0$
(I3) $\mathcal{J}(\wedge)(a,b) = \min\{a,b\}$
(I4) $\mathcal{J}(\vee)(a,b) = \max\{a,b\}$
(I5) $\mathcal{J}(\neg)(a) = 1 - a$
(I6) $\mathcal{J}(\rightarrow)(a,b) = \max\{1-a, b\}$
(I7) $\mathcal{J}(\approx, \mathcal{D})(c,d) = 1$ if $c = d$ and 0 otherwise
(I8) $\mathcal{J}(\not\approx, \mathcal{D})(c,d) = 0$ if $c = d$ and 1 otherwise
(I9) $\mathcal{J}(\forall, \mathcal{D})(f) = \min\{f(a) \mid a \in \mathcal{D}\}$
(I10) $\mathcal{J}(\exists, \mathcal{D})(f) = \max\{f(a) \mid a \in \mathcal{D}\}$
(I11) $f(\mathcal{J}(\varepsilon, \mathcal{D})(f)) = \max\{f(a) \mid a \in \mathcal{D}\}$

for all $a, b \in \{0,1\}$, $\mathcal{D} \in \mathcal{U}$, $c, d \in \mathcal{D}$, and $f \in \mathcal{J}_{\mathsf{ty}}(\to)(\mathcal{D}, \{0,1\})$.

The comprehension principle states that every function designated by a λ-expression is contained in the corresponding domain. Loosely following Fitting [15, Sect. 2.4], we initially allow λ-expressions to designate arbitrary elements of the domain, to be able to define the denotation of a term. We impose restrictions afterwards using the notion of a proper interpretation, enforcing comprehension.

A *λ-designation function* $\mathcal{L}$ for a type interpretation $\mathcal{I}_{\mathsf{ty}}$ is a function that maps a valuation ξ and a λ-expression of type τ to elements of $[\![\tau]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}}$. A type interpretation, an interpretation function, and a λ-designation function form an *interpretation* $\mathcal{I} = (\mathcal{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$.

For an interpretation $\mathcal{I}$ and a valuation ξ, the denotation of a term is defined as $[\![x]\!]^{\xi}_{\mathcal{I}} = \xi(x)$, $[\![\mathsf{f}\langle\bar{\tau}_m\rangle]\!]^{\xi}_{\mathcal{I}} = \mathcal{J}(\mathsf{f}, [\![\bar{\tau}_m]\!]^{\xi}_{\mathcal{I}_{\mathsf{ty}}})$, $[\![s\,t]\!]^{\xi}_{\mathcal{I}} = [\![s]\!]^{\xi}_{\mathcal{I}}([\![t]\!]^{\xi}_{\mathcal{I}})$, and $[\![\lambda x.\, t]\!]^{\xi}_{\mathcal{I}} = \mathcal{L}(\xi, \lambda x.\, t)$. For ground terms t, the denotation does not depend on the choice of the valuation ξ, which is why we sometimes write $[\![t]\!]_{\mathcal{I}}$ for $[\![t]\!]^{\xi}_{\mathcal{I}}$.

An interpretation $\mathcal{I}$ is *proper* if $[\![\lambda x.\,t]\!]_{\mathcal{I}}^{\xi}(a) = [\![t]\!]_{\mathcal{I}}^{\xi[x\mapsto a]}$ for all λ-expressions $\lambda x.\,t$ and all valuations ξ. If a type interpretation $\mathcal{I}_{\mathsf{ty}}$ and a interpretation function $\mathcal{J}$ can be extended by a λ-designation function $\mathcal{L}$ to a proper interpretation $(\mathcal{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$, then this $\mathcal{L}$ is unique [15, Proposition 2.18]. Given an interpretation $\mathcal{I}$ and a valuation ξ, an equation $s \approx t$ is true if $[\![s]\!]_{\mathcal{I}}^{\xi}$ and $[\![t]\!]_{\mathcal{I}}^{\xi}$ are equal and it is false otherwise. A clause is true if at least one of its literals is true. A clause set is true if all its clauses are true. A proper interpretation $\mathcal{I}$ is a *model* of a clause set N, written $\mathcal{I} \models N$, if N is true in $\mathcal{I}$ for all valuations ξ.

Skolem-Aware Interpretations. Some of the rules in our calculus introduce Skolem symbols—i.e., symbols representing objects mandated by existential quantification. We define a Skolem-extended signature that contains all Skolem symbols that could possibly be needed by the calculus rules.

Definition 2. Given a term signature Σ, let the *Skolem-extended term signature* Σ_{sk} the smallest signature that contains all symbols from Σ and a symbol $\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,t\,z} : \Pi\bar{\alpha}.\,\bar{\tau} \to \upsilon$ for all types υ, variables $z : \upsilon$, terms $t : \upsilon \to o$ over the signature $(\Sigma_{\mathsf{ty}}, \Sigma_{\mathsf{sk}})$, where $\bar{\alpha}$ are the free type variables occurring in t and $\bar{x} : \bar{\tau}$ are the free term variables occurring in t in order of first occurrence.

Interpretations as defined above can interpret the Skolem symbols arbitrarily. For example, an interpretation $\mathcal{I}$ does not necessarily interpret the symbol $\mathsf{sk}_{\exists z.\,z \mathbin{\boldsymbol{\approx}} \mathsf{a}}$ as $[\![\mathsf{a}]\!]_{\mathcal{I}}$. Therefore, an inference producing $(\mathsf{sk}_{\exists z.\,z \mathbin{\boldsymbol{\approx}} \mathsf{a}} \mathbin{\boldsymbol{\approx}} \mathsf{a}) \approx \top$ from $\exists\langle\iota\rangle\,(\lambda z.\,z \mathbin{\boldsymbol{\approx}} \mathsf{a}) \approx \top$ is unsound w.r.t. $\models$. As a remedy, we define Skolem-aware interpretations as follows:

Definition 3. We call a proper interpretation over a Skolem-extended signature *Skolem-aware* if for all Skolem symbols $\mathcal{I} \models (\exists\langle\upsilon\rangle\,(\lambda z.\,t\,z)) \approx t\,(\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,t\,z}\langle\bar{\alpha}\rangle\,\bar{x})$, where $\bar{\alpha}$ are the free type variables and $\bar{x}$ are the free term variables occurring in t in order of first occurrence. An interpretation is a *Skolem-aware model* of a clause set N, written $\mathcal{I} \mathrel{|\!\!\approx} N$, if $\mathcal{I}$ is Skolem-aware and $\mathcal{I} \models N$.

3 The Calculus

The oλSup calculus closely resembles λSup, augmented with rules for Boolean reasoning that are inspired by oSup. As in λSup, superposition-like inferences are restricted to certain first-order-like subterms, the green subterms:

Definition 4 (Green subterms and positions). A *green position* of a λ-term is a finite sequence of natural numbers defined inductively as follows. For any λ-term t, the empty sequence ε is a green position of t. For all symbols $\mathsf{f} \in \Sigma \setminus \{\forall, \exists\}$, types $\bar{\tau}$, and λ-terms $\bar{u}$, if p is a green position of u_i, then $i.p$ is a green position of $\mathsf{f}\langle\bar{\tau}\rangle\,\bar{u}$.

The *green subterm* of a λ-term at a given green position is defined inductively as follows. For any λ-term t, t itself is the green subterm of t at green position ε. For all symbols $\mathsf{f} \in \Sigma \setminus \{\forall, \exists\}$, types $\bar{\tau}$, and λ-terms $\bar{u}$, if t is a green subterm of u_i at some green position p for some i, then t is the green subterm of $\mathsf{f}\langle\bar{\tau}\rangle\,\bar{u}$ at green position $i.p$.

For positions in clauses, natural numbers are not appropriate because clauses and literals are unordered. A green position in a clause C is a tuple $L.s.p$ where $L = s \approx t$ is a literal in C and p is a green position in s. The green subterm of C at position $L.s.p$ is the green subterm of s at position p.

We write $s|_p$ to denote the green subterm at position p in s. the position p may be omitted in this notation. A position p is *at or below* a position q if q is a prefix of p. A position p is *below* a position q if q is a proper prefix of p.

For example, the green subterms of $\mathsf{f}\,(\mathsf{g}\,(\neg\,\mathsf{p}))\,(\forall\langle\tau\rangle\,(\lambda x.\,\mathsf{q}))\,(y\,\mathsf{a})\,(\lambda x.\,\mathsf{h}\,\mathsf{b})$ are the term itself, $\mathsf{g}\,(\neg\,\mathsf{p})$, $\neg\,\mathsf{p}$, p, $\forall\langle\tau\rangle\,(\lambda x.\,\mathsf{q})$, $y\,\mathsf{a}$, and $\lambda x.\,\mathsf{h}\,\mathsf{b}$.

Definition 5 (Green contexts). We write $s\langle\!\langle u\rangle\!\rangle_p$ to denote a λ-term s with the green subterm u at position p and call $s\langle\!\langle\ \rangle\!\rangle_p$ a *green context*; We omit the subscript p if there are no ambiguities.

The notions of green positions, subterms, and context are lifted to $\beta\eta$-equivalence classes via the $\beta\eta\mathsf{Q}_\eta$-normal representative.

3.1 Preprocessing

Our completeness theorem requires that quantified variables do not appear in certain higher-order contexts. We use preprocessing to eliminate problematic occurrences of quantifiers.

Definition 6. The rewrite rules $\forall_\approx$ and $\exists_\approx$, which we collectively denote by $\mathsf{Q}_\approx$, are defined on λ-terms as

$$\forall\langle\tau\rangle \longrightarrow_{\forall_\approx} \lambda y.\, y \approx (\lambda x.\,\top) \qquad\qquad \exists\langle\tau\rangle \longrightarrow_{\exists_\approx} \lambda y.\, y \not\approx (\lambda x.\,\bot)$$

where the rewritten occurrence of $\mathsf{Q}\langle\tau\rangle$ is unapplied, has an argument that is not a λ-expression, or has an argument of the form $\lambda x.\,v$ such that x occurs in a nongreen position of v.

If either of these rewrite rules can be applied to a given term, the term is $\mathsf{Q}_\approx$-*reducible*; otherwise, it is $\mathsf{Q}_\approx$-*normal*. We lift this notion to $\beta\eta$-equivalence classes via the $\beta\eta\mathsf{Q}_\eta$-normal representative. A clause or clause set is $\mathsf{Q}_\approx$-normal if all contained terms are $\mathsf{Q}_\approx$-normal.

For example, the term $\lambda y.\,\exists\langle\iota\to\iota\rangle\,(\lambda x.\,\mathsf{g}\,x\,y\,(z\,y)\,(\mathsf{f}\,x))$ is $\mathsf{Q}_\approx$-normal. A term may be $\mathsf{Q}_\approx$-reducible because a quantifier appears unapplied (e.g., $\mathsf{g}\,\exists\langle\iota\rangle$); a quantified variable occurs applied (e.g., $\exists\langle\iota\to\iota\rangle\,(\lambda x.\,x\,\mathsf{a})$); a quantified variable occurs inside a nested λ-expression (e.g., $\forall\langle\iota\rangle\,(\lambda x.\,\mathsf{f}\,(\lambda y.\,x))$); or a quantified variable occurs in the argument of a variable, either a free variable (e.g., $\forall\langle\iota\rangle\,(\lambda x.\,z\,x)$) or a variable bound above the quantifier (e.g., $\lambda y.\,\exists\langle\iota\rangle\,(\lambda x.\,y\,x)$).

We can also characterize $\mathsf{Q}_\approx$-normality as follows:

Lemma 7. *Let t be a term with spine notation $t = s\,\bar{u}_n$. Then t is $\mathsf{Q}_\approx$-normal if and only if $\bar{u}_n$ are $\mathsf{Q}_\approx$-normal and*

(i) s is of the form $\mathsf{Q}\langle\tau\rangle$, $n = 1$, and u_1 is of the form $\lambda y.\,u'$ such that y occurs only in green positions of u'; or
(ii) s is a λ-expression whose body is $\mathsf{Q}_\approx$-normal; or

(iii) s is neither of the form $\mathsf{Q}\langle\tau\rangle$ nor a λ-expression.

Proof. This follows directly from Definition 6. □

In the following lemmas, our goal is to show that $\mathsf{Q}_{\approx}$-normality is invariant under $\beta\eta\mathsf{Q}_\eta$-normalization—i.e., if a λ-term t is $\mathsf{Q}_{\approx}$-normal, then so is $t{\downarrow}_{\beta\eta\mathsf{Q}_\eta}$. However, $\mathsf{Q}_{\approx}$-normality is not invariant under arbitrary $\beta\eta$-conversions. Clearly, a β-expansion can easily introduce $\mathsf{Q}_{\approx}$-reducible terms, e.g., $\mathsf{c} \longleftarrow_\beta (\lambda x.\, \mathsf{c})\,(\forall\langle\iota\rangle)$.

Lemma 8. *If t and v are $\mathsf{Q}_{\approx}$-normal λ-terms, then $t\,v$ is a $\mathsf{Q}_{\approx}$-normal λ-term.*

Proof. We prove this by induction on the structure of t. Let $s\,\bar{u}_n = t$ be the spine notation of t. By Lemma 7, $\bar{u}_n$ are $\mathsf{Q}_{\approx}$-normal and one of the lemma's three cases applies. Since t is of functional type and $\mathsf{Q}_{\approx}$-normal, s cannot be of the form $\mathsf{Q}\langle\tau\rangle$, excluding case (i). Cases (ii) and (iii) are independent of $\bar{u}_n$, and hence appending v to that tuple does not affect the $\mathsf{Q}_{\approx}$-normality of t. □

Lemma 9. *If t is a $\mathsf{Q}_{\approx}$-normal λ-term and ρ is a substitution such that $x\rho$ is $\mathsf{Q}_{\approx}$-normal for all x, then $t\rho$ is $\mathsf{Q}_{\approx}$-normal.*

Proof. We prove this by induction on the structure of t. Let $s\,\bar{u}_n = t$ be its spine notation. Since t is $\mathsf{Q}_{\approx}$-normal, by Lemma 7, u_n are $\mathsf{Q}_{\approx}$-normal and one of the following cases applies:

Case (i): s is of the form $\mathsf{Q}\langle\tau\rangle$, $n = 1$, and u_1 is of the form $\lambda y.\, u'$ such that y occurs only in green positions of u'. Since our substitutions avoid capture, $y\rho = y$ and y does not appear in $x\rho$ for all other variables x. It is clear from the definition of green positions that since y occurs only in green positions of u', y also occurs only in green positions of $u'\rho$. Moreover, by the induction hypothesis, $u_1\rho$ is $\mathsf{Q}_{\approx}$-normal. Hence, $t\rho = \mathsf{Q}\langle\tau\rho\rangle\,(u_1\rho) = \mathsf{Q}\langle\tau\rho\rangle\,(\lambda y.\, u'\rho)$ is also $\mathsf{Q}_{\approx}$-normal.

Case (ii): s is a λ-expression whose body is $\mathsf{Q}_{\approx}$-normal. Then $t\rho = (\lambda y.\, s'\rho)\,(\bar{u}_n\rho)$ for some $\mathsf{Q}_{\approx}$-normal λ-term s'. By the induction hypothesis, $s'\rho$ and $\bar{u}_n\rho$ are $\mathsf{Q}_{\approx}$-normal. Therefore, $t\rho$ is also $\mathsf{Q}_{\approx}$-normal.

Case (iii): s is neither of the form $\mathsf{Q}\langle\tau\rangle$ nor a λ-expression. If s is of the form $\mathsf{f}\langle\bar{\tau}\rangle$ for some $\mathsf{f} \notin \{\forall, \exists\}$, then $t\rho = \mathsf{f}\langle\bar{\tau}\rho\rangle\,(\bar{u}_n\rho)$. By the induction hypothesis, $\bar{u}_n\rho$ are $\mathsf{Q}_{\approx}$-normal, and therefore $t\rho$ is also $\mathsf{Q}_{\approx}$-normal. Otherwise, s is a variable x and hence $t\rho = x\rho\,(\bar{u}_n\rho)$. Since $x\rho$ is $\mathsf{Q}_{\approx}$-normal by assumption and $\bar{u}_n\rho$ are $\mathsf{Q}_{\approx}$-normal by the induction hypothesis, it follows from (repeated application of) Lemma 8 that $t\rho$ is also $\mathsf{Q}_{\approx}$-normal. □

Lemma 10. *Let t be a λ-term of functional type that does not contain the variable x. If $\lambda x.\, t\,x$ is $\mathsf{Q}_{\approx}$-normal, then t is $\mathsf{Q}_{\approx}$-normal.*

Proof. Since $\lambda x.\, t\,x$ is $\mathsf{Q}_{\approx}$-normal, $t\,x$ is also $\mathsf{Q}_{\approx}$-normal. Let $s\,\bar{u}_n = t$. Since $t\,x$ is $\mathsf{Q}_{\approx}$-normal and x is not a λ-expression, s cannot be a quantifier by Lemma 7. Cases (ii) and (iii) are independent of $\bar{u}_n$, and hence removing x from that tuple does not affect $\mathsf{Q}_{\approx}$-normality. Thus, $t\,x$ being $\mathsf{Q}_{\approx}$-normal implies that t is $\mathsf{Q}_{\approx}$-normal. □

Lemma 11. *Let t be a λ-term and x a variable occurring only in green positions of t. Let t' be a term obtained via a $\beta\eta\mathsf{Q}_\eta$-normalization step from t. Then x occurs only in green positions of t'.*

Proof. By induction on the structure of t. If x does not occur in t, the claim is obvious. If $t = x$, there is no possible $\beta\eta\mathsf{Q}_\eta$-normalization step because for these steps the head of the rewritten term must be either a λ-expression or a quantifier. So we now assume that x does occur in t and that $t \neq x$. Then, by the assumption that x occurs only in green positions, t must be of the form $\mathsf{f}\langle\bar{\tau}\rangle\,\bar{u}$ for some $\mathsf{f} \in \Sigma \setminus \{\forall,\exists\}$, some types $\bar{\tau}$ and some λ-terms $\bar{u}$. The $\beta\eta\mathsf{Q}_\eta$-normalization step must take place in one of the $\bar{u}$, yielding $\bar{u}'$ such that $t' = \mathsf{f}\langle\bar{\tau}\rangle\,\bar{u}'$. By the induction hypothesis, x occurs only in green positions of $\bar{u}'$ and therefore only in green positions of t'. □

Lemma 12. *Let t be $\mathsf{Q}_\approx$-normal and let t' be obtained from t by a $\beta\eta\mathsf{Q}_\eta$-normalization step. If it is an η-reduction step, we assume that it happens not directly below a quantifier. Then t' is also $\mathsf{Q}_\approx$-normal.*

Proof. Let $s\,\bar{u}_n = t$. By Lemma 7, $\bar{u}_n$ are $\mathsf{Q}_\approx$-normal, and one of the following cases applies:

Case (i): s is of the form $\mathsf{Q}\langle\tau\rangle$, $n = 1$, and u_1 is of the form $\lambda y.\,v$ such that y occurs only in green positions of v. Then the normalization cannot happen at t, because s is of the form $\mathsf{Q}\langle\tau\rangle$ and u_1 is a λ-expression already. It cannot happen at u_1 by the assumption of this lemma. So it must happen in v, yielding some λ-term v'. Then $t = s\,(\lambda x.\,v')$. The λ-term v' is $\mathsf{Q}_\approx$-normal by the induction hypothesis and hence $(\lambda x.\,v')$ is $\mathsf{Q}_\approx$-normal. Since x occurs only in green positions of v, by Lemma 11, x occurs only in green positions of v'. Thus, t' is $\mathsf{Q}_\approx$-normal.

Cases (ii) and (iii): s is a λ-expression whose body is $\mathsf{Q}_\approx$-normal; or s is neither of the form $\mathsf{Q}\langle\tau\rangle$ nor a λ-expression.

If the $\beta\eta\mathsf{Q}_\eta$-normalization step happens in some u_i, yielding some λ-term u_i', then u_i' is $\mathsf{Q}_\approx$-normal by the induction hypothesis. Thus, $t' = s\,u_1\,\cdots\,u_{i-1}\,u_i'\,u_{i+1}\cdots\,u_n$ is also $\mathsf{Q}_\approx$-normal.

Otherwise, if $s = \lambda x.\,v$ and the $\beta\eta\mathsf{Q}_\eta$-normalization step happens in v, yielding some λ-term v', then v' is $\mathsf{Q}_\approx$-normal by the induction hypothesis. Thus, $t' = (\lambda x.\,v')\,\bar{u}_n$ is also $\mathsf{Q}_\approx$-normal.

Otherwise, the $\beta\eta\mathsf{Q}_\eta$-normalization step happens at $s\,\bar{u}_m$ for some $m \leq n$, yielding some λ-term v'. Then $t' = v'\,u_{m+1}\,\cdots\,u_n$. The λ-terms s and $\bar{u}_m$ are $\mathsf{Q}_\approx$-normal and by repeated application of Lemma 8, $s\,\bar{u}_m$ is also $\mathsf{Q}_\approx$-normal. The λ-term v' is $\mathsf{Q}_\approx$-normal by Lemma 9 (for β-reductions) or Lemma 10 (for η-reductions). The normalization step cannot be a Q_η-normalization because s is not a quantifier. Since $\bar{u}_n$ are also $\mathsf{Q}_\approx$-normal, by repeated application of Lemma 8, $t[v'] = v'\,u_{m+1}\,\cdots\,u_n$ is also $\mathsf{Q}_\approx$-normal. □

A direct consequence of this lemma is that $\mathsf{Q}_\approx$-normality is invariant under $\beta\eta\mathsf{Q}_\eta$-normalization, as we wanted to show:

Corollary 13. *If t is a $\mathsf{Q}_\approx$-normal λ-term, then $t{\downarrow}_{\beta\eta\mathsf{Q}_\eta}$ is also $\mathsf{Q}_\approx$-normal.*

As mentioned above, the converse does not hold. Therefore, following our convention, $\mathsf{Q}_\approx$-normality is defined on terms (i.e., $\beta\eta$-equivalence classes) via $\beta\eta\mathsf{Q}_\eta$-normal forms. It follows that $\mathsf{Q}_\approx$-normality is well-behaved under applications of terms as well:

Lemma 14. *If t and v are $\mathsf{Q}_\approx$-normal terms where t is of functional type, then $t\,v$ is also $\mathsf{Q}_\approx$-normal.*

Proof. Since $\mathsf{Q}_{\approx}$-normality is defined via $\beta\eta\mathsf{Q}_{\eta}$-normal forms, we must show that if $t{\downarrow}_{\beta\eta\mathsf{Q}_{\eta}}$ and $v{\downarrow}_{\beta\eta\mathsf{Q}_{\eta}}$ are $\mathsf{Q}_{\approx}$-normal, then $t\,v{\downarrow}_{\beta\eta\mathsf{Q}_{\eta}}$ is $\mathsf{Q}_{\approx}$-normal. By Lemma 8, the λ-term $(t{\downarrow}_{\beta\eta\mathsf{Q}_{\eta}})\,(v{\downarrow}_{\beta\eta\mathsf{Q}_{\eta}})$ is $\mathsf{Q}_{\approx}$-normal. By Corollary 13, $((t{\downarrow}_{\beta\eta\mathsf{Q}_{\eta}})\,(v{\downarrow}_{\beta\eta\mathsf{Q}_{\eta}})){\downarrow}_{\beta\eta\mathsf{Q}_{\eta}} = (t\,v){\downarrow}_{\beta\eta\mathsf{Q}_{\eta}}$ is $\mathsf{Q}_{\approx}$-normal. □

A preprocessor $\mathsf{Q}_{\approx}$-normalizes the input problem. It clearly terminates because each $\mathsf{Q}_{\approx}$-step reduces the number of quantifiers. The $\mathsf{Q}_{\approx}$-normality of the initial clause set of a derivation will be a precondition of the completeness theorem. Although inferences may produce $\mathsf{Q}_{\approx}$-reducible clauses, we do not $\mathsf{Q}_{\approx}$-normalize during the derivation process itself. Instead, $\mathsf{Q}_{\approx}$-reducible ground instances of clauses will be considered redundant by the redundancy criterion. Thus, clauses whose ground instances are all $\mathsf{Q}_{\approx}$-reducible can be deleted. However, there are $\mathsf{Q}_{\approx}$-reducible clauses, such as $x\forall\langle\iota\rangle \approx \mathsf{a}$, that nevertheless have $\mathsf{Q}_{\approx}$-normal ground instances. Such clauses must be kept because the completeness proof relies on their $\mathsf{Q}_{\approx}$-normal ground instances.

In principle, we could omit the side condition of the $\mathsf{Q}_{\approx}$-rewrite rules and eliminate all quantifiers. However, the calculus (especially, the redundancy criterion) performs better with quantifiers than with λ-expressions, which is why we restrict $\mathsf{Q}_{\approx}$-normalization as much as the completeness proof allows. Extending the preprocessing to eliminate all Boolean terms as in Kotelnikov et al. [24] does not work for higher-order logic because Boolean terms can contain variables bound by enclosing λ-expressions.

3.2 Term Orders and Selection Functions

The calculus is parameterized by a strict and a nonstrict term order, a literal selection function, and a Boolean subterm selection function. These concepts are defined below.

Definition 15 (Strict ground term order). A well-founded strict total order $\succ$ on ground terms is a *strict ground term order* if it satisfies the following criteria, where $\succeq$ denotes the reflexive closure of $\succ$:

(O1) compatibility with green contexts: $s' \succ s$ implies $t\langle\!\langle s'\rangle\!\rangle \succ t\langle\!\langle s\rangle\!\rangle$;
(O2) green subterm property: $t\langle\!\langle s\rangle\!\rangle \succeq s$;
(O3) $u \succ \bot \succ \top$ for all terms $u \neq \top, \bot$;
(O4) $\mathsf{Q}\langle\tau\rangle\, t \succ t\, u$ for all types τ, terms t, and terms u such that $\mathsf{Q}\langle\tau\rangle\, t$ and u are $\mathsf{Q}_{\approx}$-normal and the only Boolean green subterms of u are $\top$ and $\bot$.

Given a strict ground term order, we extend it to literals and clauses via the multiset extensions in the standard way [2, Sect. 2.4].

Remark 16. The property $\mathsf{Q}\langle\tau\rangle\, t \succ t\, u$ from (O4) cannot be achieved in general—a fact that Christoph Benzmüller made us aware about. Without further restrictions, it would imply that $\mathsf{Q}\langle\tau\rangle(\lambda x.\, x) \succ (\lambda x.\, x)(\mathsf{Q}\langle\tau\rangle(\lambda x.\, x)) = \mathsf{Q}\langle\tau\rangle(\lambda x.\, x)$, contradicting irreflexivity of $\succ$. Restricting the Boolean green subterms of u to be only $\top$ and $\bot$ resolves this issue. A second issue is that (O4) without further restrictions would imply $\mathsf{Q}\langle\tau\rangle\,(\lambda y.\, y\,\mathsf{a}) \succ (\lambda y.\, y\,\mathsf{a})(\lambda x.\, \mathsf{Q}\langle\tau\rangle\,(\lambda y.\, y\,\mathsf{a})) = \mathsf{Q}\langle\tau\rangle\,(\lambda y.\, y\,\mathsf{a})$, again contradicting irreflexivity. The restriction on the Boolean green subterms of u does not apply here because the Boolean subterms of $\lambda x.\, \mathsf{Q}\langle\tau\rangle\,(\lambda y.\, y\,\mathsf{a})$ are not green. The restriction to $\mathsf{Q}_{\approx}$-normal terms resolves this second issue, but it forces us to preprocess the input problem.

Definition 17 (Strict term order). A *strict term order* is a relation $\succ$ on terms, literals, and clauses such that its restriction to ground entities is a strict ground term order and such that it is stable under grounding substitutions (i.e., $t \succ s$ implies $t\theta \succ s\theta$ for all substitutions θ grounding the entities t and s).

Definition 18 (Nonstrict term order). Given a strict term order $\succ$ and its reflexive closure $\succeq$, a *nonstrict term order* is a relation $\succsim$ on terms, literals, and clauses such that $t \succsim s$ implies $t\theta \succeq s\theta$ for all θ grounding the entities t and s.

Although we call them orders, a strict term order $\succ$ is not required to be transitive on nonground entities, and a nonstrict term order $\succsim$ does not need to be transitive at all. Normally, $t \succeq s$ should imply $t \succsim s$, but this is not required either. A nonstrict term order $\succsim$ allows us to be more precise than the reflexive closure $\succeq$ of $\succ$. For example, we cannot have $y\,\mathsf{b} \succeq y\,\mathsf{a}$, because $y\,\mathsf{b} \neq y\,\mathsf{a}$ and $y\,\mathsf{b} \not\succ y\,\mathsf{a}$ by stability under grounding substitutions (with $\{y \mapsto \lambda x.\, \mathsf{c}\}$). But we can have $y\,\mathsf{b} \succsim y\,\mathsf{a}$ if $\mathsf{b} \succ \mathsf{a}$. In practice, the strict and the nonstrict term order should be chosen so that they can compare as many pairs of terms as possible while being computable and reasonably efficient.

Definition 19 (Maximality). An element x of a multiset M is $\trianglerighteq$-*maximal* for some relation $\trianglerighteq$ if for all $y \in M$ with $y \trianglerighteq x$, we have $y = x$. It is *strictly* $\trianglerighteq$-maximal if it is $\trianglerighteq$-maximal and occurs only once in M.

Definition 20 (Literal selection function). A literal selection function is a mapping from each clause to a subset of its literals. The literals in this subset are called *selected*. The following restrictions apply:

- A literal must not be selected if it is positive and neither side is $\bot$.
- A literal $L\langle y\rangle$ must not be selected if $y\,\bar{u}_n$, with $n \geq 1$, is a $\succeq$-maximal term of the clause.

Definition 21 (Boolean subterm selection function). A Boolean subterm selection function is a function mapping each clause C to a subset of the green positions with Boolean subterms in C. The positions in this subset are called *selected* in C. Informally, we also say that the Boolean subterms at these positions are selected. The following restrictions apply:

- A subterm $s\langle y\rangle$ must not be selected if $y\,\bar{u}_n$, with $n \geq 1$, is a $\succeq$-maximal term of the clause.
- A subterm must not be selected if it is $\top$ or $\bot$ or a variable-headed term.
- A subterm must not be selected if it is at the topmost position on either side of a positive literal.

3.3 The Core Inference Rules

Let $\succ$ be a strict term order, let $\succsim$ be a nonstrict term order, let *HLitSel* be a literal selection function, and let *HBoolSel* be a Boolean subterm selection function. The calculus rules depend on the following auxiliary notions.

Definition 22 (Eligibility). A literal L is (*strictly*) $\trianglerighteq$-*eligible* w.r.t. a substitution σ in C for some relation $\trianglerighteq$ if it is selected in C or there are no selected literals and no selected Boolean subterms in C and $L\sigma$ is (strictly) $\trianglerighteq$-maximal in $C\sigma$.

The $\trianglerighteq$-eligible positions of a clause C w.r.t. a substitution σ are inductively defined as follows:

(E1) Any selected position is $\trianglerighteq$-eligible.
(E2) If a literal $L = s \mathrel{\dot{\approx}} t$ with $s\sigma \not\trianglelefteq t\sigma$ is either $\trianglerighteq$-eligible and negative or strictly $\trianglerighteq$-eligible and positive, then $L.s.\varepsilon$ is $\trianglerighteq$-eligible.
(E3) If the position p is $\trianglerighteq$-eligible and the head of $C|_p$ is not $\boldsymbol{\approx}$ or $\boldsymbol{\not\approx}$, the positions of all direct green subterms are $\trianglerighteq$-eligible.
(E4) If the position p is $\trianglerighteq$-eligible and $C|_p$ is of the form $s \boldsymbol{\approx} t$ or $s \boldsymbol{\not\approx} t$, then the position of s is $\trianglerighteq$-eligible if $s\sigma \not\trianglelefteq t\sigma$ and the position of t is $\trianglerighteq$-eligible if $s\sigma \not\trianglerighteq t\sigma$.

If σ is the identity substitution, we leave it implicit.

We define deeply occurring variables as in λSup, but exclude λ-expressions directly below quantifiers:

Definition 23 (Deep occurrences). A variable *occurs deeply* in a clause C if it occurs inside an argument of an applied variable or inside a λ-expression that is not directly below a quantifier.

For example, x and z occur deeply in $\mathsf{f}\,x\,y \approx y\,x \lor z \not\approx (\lambda w.\, z\,\mathsf{a}) \lor \forall\langle\iota\rangle\,(\lambda u.\,\mathsf{p}\,y) \approx \top$, whereas y does not occur deeply.

Fluid terms are defined as in λSup, using the $\beta\eta\mathsf{Q}_\eta$-normal form:

Definition 24 (Fluid terms). A term t is called *fluid* if (1) $t{\downarrow_{\beta\eta\mathsf{Q}_\eta}}$ is of the form $y\,\bar{u}_n$ where $n \geq 1$, or (2) $t{\downarrow_{\beta\eta\mathsf{Q}_\eta}}$ is a λ-expression and there exists a substitution σ such that $t\sigma{\downarrow_{\beta\eta\mathsf{Q}_\eta}}$ is not a λ-expression (due to η-reduction).

Case (2) can arise only if t contains an applied variable. Intuitively, fluid terms are terms whose η-short β-normal form can change radically as a result of instantiation. For example, $\lambda x.\, y\,\mathsf{a}\,(z\,x)$ is fluid because applying $\{z \mapsto \lambda x.\, x\}$ makes the λ vanish: $(\lambda x.\, y\,\mathsf{a}\,x) = y\,\mathsf{a}$. Similarly, $\lambda x.\, \mathsf{f}\,(y\,x)\,x$ is fluid because $(\lambda x.\, \mathsf{f}\,(y\,x)\,x)\{y \mapsto \lambda x.\, \mathsf{a}\} = (\lambda x.\, \mathsf{f}\,\mathsf{a}\,x) = \mathsf{f}\,\mathsf{a}$.

The rules of our calculus are stated as follows. The superposition rule strongly resembles the one of λSup but uses our new notion of eligibility, and the new conditions 9 and 10 stem from the SUP rule of oSup:

$$\frac{\overbrace{D' \lor t \approx t'}^{D} \qquad C\langle u\rangle}{(D' \lor C\langle t'\rangle)\sigma}\ \textsc{Sup}$$

1. u is not fluid; 2. u is not a variable deeply occurring in C;
3. *variable condition*: if u is a variable y, there must exist a grounding substitution θ such that $t\sigma\theta \succ t'\sigma\theta$ and $C\sigma\theta \prec C''\sigma\theta$, where $C'' = C\{y \mapsto t'\}$;
4. $\sigma \in \mathrm{CSU}(t,u)$; 5. $t\sigma \not\precsim t'\sigma$; 6. the position of u is $\succsim$-eligible in C w.r.t. σ;
7. $C\sigma \not\precsim D\sigma$; 8. $t \approx t'$ is strictly $\succsim$-eligible in D w.r.t. σ;

9. $t\sigma$ is not a fully applied logical symbol;
10. if $t'\sigma = \bot$, the position of the subterm u is at the top level of a positive literal.

The second rule is a variant of SUP that focuses on fluid green subterms. It stems from λSup.

$$\frac{\overbrace{D' \vee t \approx t'}^{D} \quad C\langle u\rangle}{(D' \vee C\langle z\,t'\rangle)\sigma}\ \textsc{FluidSup}$$

with the following side conditions, in addition to SUP's conditions 5 to 10:

1. u is a variable deeply occurring in C or u is fluid;
2. z is a fresh variable; 3. $\sigma \in \mathrm{CSU}(z\,t, u)$; 4. $(z\,t')\sigma \neq (z\,t)\sigma$.

The ERES and EFACT rules are copied from λSup. As a minor optimization, we add a condition to EFACT that nothing is selected, which is only necessary because positive literals of the form $u \approx \top$ can be selected.

$$\frac{\overbrace{C' \vee u \not\approx u'}^{C}}{C'\sigma}\ \textsc{ERes} \qquad \frac{\overbrace{C' \vee u' \approx v' \vee u \approx v}^{C}}{(C' \vee v \not\approx v' \vee u \approx v')\sigma}\ \textsc{EFact}$$

For ERES: $\sigma \in \mathrm{CSU}(u,u')$ and $u \not\approx u'$ is $\succsim$-eligible in C w.r.t. σ. For EFACT: $\sigma \in \mathrm{CSU}(u,u')$, $u\sigma \not\precsim v\sigma$, $(u \approx v)\sigma$ is $\succsim$-maximal in $C\sigma$, and nothing is selected in C.

Argument congruence—the property that $t \approx s$ entails $t\,z \approx s\,z$—is embodied by the rule ARGCONG, which is identical with the rule of λSup:

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

where $n > 0$ and σ is the most general type substitution that ensures well-typedness of the conclusion. In particular, if s accepts k arguments, then ARGCONG will yield k conclusions—one for each $n \in \{1,\ldots,k\}$—where σ is the identity substitution. If the result type of s is a type variable, ARGCONG will yield infinitely many additional conclusions—one for each $n > k$—where σ instantiates the result type of s with $\alpha_1 \to \cdots \to \alpha_{n-k} \to \beta$ for fresh $\bar{\alpha}_{n-k}$ and β. Moreover, the literal $s \approx s'$ must be strictly $\succsim$-eligible in C w.r.t. σ, and $\bar{x}_n$ is a tuple of distinct fresh variables.

The following rules are concerned with Boolean reasoning and originate from oSup. They have been adapted to support polymorphism and applied variables.

$$\frac{C\langle u\rangle}{(C\langle\bot\rangle \vee u \approx \top)\sigma}\ \textsc{BoolHoist}$$

1. σ is a type unifier of the type of u with the Boolean type o (i.e., the identity if u is Boolean or $\alpha \mapsto \mathrm{o}$ if u is of type α for some type variable α);
2. u is neither variable-headed nor a fully applied logical symbol;

3. the position of u is $\gtrsim$-eligible in C;
4. the occurrence of u is not at the top level of a positive literal.

$$\frac{\overbrace{C' \vee s \approx s'}^{C}}{C'\sigma}\ \textsc{FalseElim}$$

1. $\sigma \in \mathrm{CSU}(s \approx s', \bot \approx \top)$; 2. $s \approx s'$ is strictly $\gtrsim$-eligible in C w.r.t. σ.

$$\frac{C\langle u \rangle}{(C\langle \bot \rangle \vee x \approx y)\sigma}\ \textsc{EqHoist} \qquad \frac{C\langle u \rangle}{(C\langle \top \rangle \vee x \approx y)\sigma}\ \textsc{NeqHoist}$$

$$\frac{C\langle u \rangle}{(C\langle \bot \rangle \vee y\, x \approx \top)\sigma}\ \textsc{ForallHoist} \qquad \frac{C\langle u \rangle}{(C\langle \top \rangle \vee y\, x \approx \bot)\sigma}\ \textsc{ExistsHoist}$$

1. $\sigma \in \mathrm{CSU}(u, x \mathbin{\approx} y)$, $\sigma \in \mathrm{CSU}(u, x \mathbin{\not\approx} y)$, $\sigma \in \mathrm{CSU}(u, \forall\langle\alpha\rangle\, y)$, or $\sigma \in \mathrm{CSU}(u, \exists\langle\alpha\rangle\, y)$, respectively;
2. x, y, and α are fresh variables; 3. the position of u is $\gtrsim$-eligible in C w.r.t. σ;
4. if the head of u is a variable, it must be applied and the affected literal must be of the form $u \approx \top$, $u \approx \bot$, or $u \approx v$ where v is a variable-headed term.

$$\frac{C\langle u \rangle}{C\langle t' \rangle\sigma}\ \textsc{BoolRw}$$

1. $\sigma \in \mathrm{CSU}(t, u)$ and (t, t') is one of the following pairs:

$$\begin{array}{cccc}
(\neg\bot, \top) & (\bot \wedge \bot, \bot) & (\bot \vee \bot, \bot) & (\bot \rightarrow \bot, \top) \\
(\neg\top, \bot) & (\top \wedge \bot, \bot) & (\top \vee \bot, \top) & (\top \rightarrow \bot, \bot) \\
(y \approx y, \top) & (\bot \wedge \top, \bot) & (\bot \vee \top, \top) & (\bot \rightarrow \top, \top) \\
(y \not\approx y, \bot) & (\top \wedge \top, \top) & (\top \vee \top, \top) & (\top \rightarrow \top, \top)
\end{array}$$

where y is a fresh variable;
2. u is not a variable; 3. the position of u is $\gtrsim$-eligible in C w.r.t. σ;
4. if the head of u is a variable, it must be applied and the affected literal must be of the form $u \approx \top$, $u \approx \bot$, or $u \approx v$ where v is a variable-headed term.

$$\frac{C\langle u \rangle}{C\langle y\,(\mathsf{sk}_{\Pi\bar\alpha.\,\forall\bar x.\,\exists z.\,\neg y\sigma\, z}\langle\bar\alpha\rangle\,\bar x)\rangle\sigma}\ \textsc{ForallRw}$$

$$\frac{C\langle u \rangle}{C\langle y\,(\mathsf{sk}_{\Pi\bar\alpha.\,\forall\bar x.\,\exists z.\,y\sigma\, z}\langle\bar\alpha\rangle\,\bar x)\rangle\sigma}\ \textsc{ExistsRw}$$

1. $\sigma \in \mathrm{CSU}(\forall\langle\beta\rangle\, y,\, u)$ and $\sigma \in \mathrm{CSU}(\exists\langle\beta\rangle\, y,\, u)$, respectively, where β is a fresh type variable, y is a fresh term variable, $\bar{\alpha}$ are the free type variables and $\bar{x}$ are the free term variables occurring in $y\sigma$ in order of first occurrence;
2. u is not a variable; 3. the position of u is $\succsim$-eligible in C w.r.t. σ;
4. if the head of u is a variable, it must be applied and the affected literal must be of the form $u \approx \top$, $u \approx \bot$, or $u \approx v$ where v is a variable-headed term;
5. for FORALLRW, the indicated occurrence of u is not in a literal $u \approx \top$, and for EXISTSRW, the indicated occurrence of u is not in a literal $u \approx \bot$.

In principle, the subscript of the Skolems above could be normalized using Boolean tautologies to share as many Skolem symbols as possible. This is an extension of our calculus that we did not investigate any further.

Like SUP, also the Boolean rules must be simulated in fluid terms. The following rules are Boolean counterparts of FLUIDSUP:

$$\frac{C\langle u\rangle}{(C\langle z\,\bot\rangle \vee x \approx \top)\sigma}\ \textsc{FluidBoolHoist}$$

1. u is fluid;
2. z and x are fresh variables; 3. $\sigma \in \mathrm{CSU}(z\,x,\, u)$; 4. $(z\,\bot)\sigma \neq (z\,x)\sigma$;
5. $x\sigma \neq \top$ and $x\sigma \neq \bot$; 6. the position of u is $\succsim$-eligible in C w.r.t. σ.

$$\frac{C\langle u\rangle}{(C\langle z\,\top\rangle \vee x \approx \bot)\sigma}\ \textsc{FluidLoobHoist}$$

Same conditions as FLUIDBOOLHOIST, but $\bot$ is replaced by $\top$ in condition 4.

In addition to the inference rules, our calculus relies on two axioms, below. Axiom (EXT), from λSup, embodies functional extensionality; the expression $\mathsf{diff}\langle\alpha,\beta\rangle$ abbreviates $\mathsf{sk}_{\Pi\alpha\beta.\,\forall z y.\,\exists x.\, z\,x \not\approx y\,x}\langle\alpha,\beta\rangle$. Axiom (CHOICE) characterizes the Hilbert choice operator ε.

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

$$y\,x \approx \bot \vee y\,(\varepsilon\langle\alpha\rangle\, y) \approx \top \qquad (\textsc{Choice})$$

3.4 Rationale for the Rules

Most of the calculus's rules are adapted from its precursors. SUP, ERES, and EFACT are already present in Sup, with slightly different side conditions. Notably, as in λfSup and λSup, SUP inferences are required only into green contexts. Other subterms are accessed indirectly via ARGCONG and (EXT).

The rules BOOLHOIST, EQHOIST, NEQHOIST, FORALLHOIST, EXISTSHOIST, FALSEELIM, BOOLRW, FORALLRW, and EXISTSRW, concerned with Boolean reasoning, stem from oSup, which was inspired by $\leftrightarrow$Sup. Except for BOOLHOIST and FALSEELIM, these rules have a condition stating that "if the head of u is a variable, it must be applied and the affected literal must be of the form $u \approx \top$, $u \approx \bot$, or $u \approx v$ where v is a variable-headed term." The inferences at variable-headed terms permitted by this condition are our form of primitive substitution [1,18], a mechanism that blindly substitutes logical connectives and quantifiers for variables z with a Boolean result type.

Example 25. Our calculus can prove that Leibniz equality implies equality (i.e., if two values behave the same for all predicates, they are equal) as follows:

$$\dfrac{\mathsf{a} \not\approx \mathsf{b} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{z\,\mathsf{a} \approx \bot \lor z\,\mathsf{b} \approx \top}{(x\,\mathsf{a} \mathbin{\boldsymbol{\approx}} y\,\mathsf{a}) \approx \bot \lor \bot \approx \top \lor x\,\mathsf{b} \approx y\,\mathsf{b}}\,\textsc{EqHoist}}{\top \approx \bot \lor \bot \approx \top \lor w\,\mathsf{a}\,\mathsf{b}\,\mathsf{b} \approx w\,\mathsf{b}\,\mathsf{a}\,\mathsf{b}}\,\textsc{BoolRw}}{\bot \approx \top \lor w\,\mathsf{a}\,\mathsf{b}\,\mathsf{b} \approx w\,\mathsf{b}\,\mathsf{a}\,\mathsf{b}}\,\textsc{FalseElim}}{w\,\mathsf{a}\,\mathsf{b}\,\mathsf{b} \approx w\,\mathsf{b}\,\mathsf{a}\,\mathsf{b}}\,\textsc{FalseElim}}{\dfrac{\mathsf{a} \not\approx \mathsf{a}}{\bot}\,\textsc{ERes}}\,\textsc{Sup}$$

The EQHOIST inference, applied on $z\,\mathsf{b}$, illustrates how our calculus introduces logical symbols without a dedicated primitive substitution rule. Although $\mathbin{\boldsymbol{\approx}}$ does not appear in the premise, we still need to apply EQHOIST on $z\,\mathsf{b}$ with $\mathrm{CSU}(z\,\mathsf{b},\, x_0 \mathbin{\boldsymbol{\approx}} y_0) = \{\{z \mapsto \lambda v.\, x\,v \mathbin{\boldsymbol{\approx}} y\,v,\ x_0 \mapsto x\,\mathsf{b},\ y_0 \mapsto y\,\mathsf{b}\}\}$. Other calculi [1, 8, 18, 30] would apply an explicit primitive substitution rule instead, yielding essentially $(x\,\mathsf{a} \mathbin{\boldsymbol{\approx}} y\,\mathsf{a}) \approx \bot \lor (x\,\mathsf{b} \mathbin{\boldsymbol{\approx}} y\,\mathsf{b}) \approx \top$. However, in our approach this clause is subsumed and could be discarded immediately. By hoisting the equality to the clausal level, we bypass the redundancy criterion.

Next, BOOLRW can be applied to $x\,\mathsf{a} \mathbin{\boldsymbol{\approx}} y\,\mathsf{a}$ with $\mathrm{CSU}(x\,\mathsf{a} \mathbin{\boldsymbol{\approx}} y\,\mathsf{a},\, y_0 \mathbin{\boldsymbol{\approx}} y_0) = \{\{x \mapsto \lambda v.\, w\,\mathsf{a}\,v\,v,\ y \mapsto \lambda v.\, w\,v\,\mathsf{a}\,v,\ y_0 \mapsto w\,\mathsf{a}\,\mathsf{a}\,\mathsf{a}\}\}$. The two FALSEELIM steps remove the $\bot \approx \top$ literals. Then SUP is applicable with the unifier $\{w \mapsto \lambda x_1\,x_2\,x_3.\, x_2\} \in \mathrm{CSU}(\mathsf{b},\, w\,\mathsf{a}\,\mathsf{b}\,\mathsf{b})$, and ERES derives the contradiction.

This mechanism resembling primitive substitution is not the only way our calculus can instantiate variables with logical symbols. Often, the correct instantiation can also be found by unification with a logical symbol that is already present:

Example 26. The following derivation shows that there exists a function y that is equivalent to the conjunction of $\mathsf{p}\,x$ and $\mathsf{q}\,x$ for all arguments x:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\exists\langle\iota\rangle\,(\lambda y.\, \forall\langle\iota\rangle\,(\lambda x.\, y\,x \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,x \land \mathsf{q}\,x))) \approx \bot}{\top \approx \bot \lor \forall\langle\iota\rangle\,(\lambda x.\, y'\,x \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,x \land \mathsf{q}\,x)) \approx \bot}\,\textsc{ExistsHoist}}{\top \approx \bot \lor (y'\,(\mathsf{sk}\,y') \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,(\mathsf{sk}\,y') \land \mathsf{q}\,(\mathsf{sk}\,y'))) \approx \bot}\,\textsc{ForallRw}}{\top \approx \bot \lor \top \approx \bot}\,\textsc{BoolRw}}{\top \approx \bot}\,\textsc{FalseElim}}{\bot}\,\textsc{FalseElim}$$

Here, sk stands for $\mathsf{sk}_{\forall u.\, \exists v.\, \neg u\,v \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,v \land \mathsf{q}\,v)}$. First, we use EXISTSHOIST to resolve the existential quantifier, using the unifier $\{\alpha \mapsto \iota,\, z \mapsto \lambda y.\, \forall\langle\iota\rangle\,(\lambda x.\, y\,x \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,x \land \mathsf{q}\,x))\} \in \mathrm{CSU}(\exists\langle\iota\rangle\,(\lambda y.\, \forall\langle\iota\rangle\,(\lambda x.\, y\,x \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,x \land \mathsf{q}\,x))),\, \exists\langle\alpha\rangle\,z)$ for fresh variables α, y', and z. Then FORALLRW skolemizes the universal quantifier, using the unifier $\{\beta \mapsto \iota,\, z' \mapsto \lambda x.\, y'\,x \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,x \land \mathsf{q}\,x)\} \in \mathrm{CSU}(\forall\langle\beta\rangle\,z',\, \forall\langle\iota\rangle\,(\lambda x.\, y'\,x \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,x \land \mathsf{q}\,x)))$ for fresh variables β and z'. The Skolem symbol takes y' as argument because it occurs free in $\lambda x.\, y'\,x \mathbin{\boldsymbol{\approx}} (\mathsf{p}\,x \land \mathsf{q}\,x)$. Then

BOOLRW applies because the terms $y'\,(\mathsf{sk}\,y')$ and $\mathsf{p}\,(\mathsf{sk}\,y') \wedge \mathsf{q}\,(\mathsf{sk}\,y')$ are unifiable and thus $y'\,(\mathsf{sk}\,y') \mathrel{\boldsymbol{\approx}} (\mathsf{p}\,(\mathsf{sk}\,y') \wedge \mathsf{q}\,(\mathsf{sk}\,y'))$ is unifiable with $y \mathrel{\boldsymbol{\approx}} y$. Finally, two FALSEELIM inferences lead to the empty clause.

Like in λSup, the FLUIDSUP rule is responsible for simulating superposition inferences below applied variables, other fluid terms, and deeply occurring variables. Complementarily, FLUIDBOOLHOIST and FLUIDLOOBHOIST simulate the various Boolean inference rules below fluid terms. Initially, we considered adding a fluid version of each rule that operates on Boolean subterms, but we discovered that FLUIDBOOLHOIST and FLUIDLOOBHOIST suffice to achieve refutational completeness.

Example 27. The following clause set demonstrates the need for the rules FLUIDBOOLHOIST and FLUIDLOOBHOIST:

$$\mathsf{h}\,(y\,\mathsf{b}) \not\approx \mathsf{h}\,(\mathsf{g}\,\bot) \vee \mathsf{h}\,(y\,\mathsf{a}) \not\approx \mathsf{h}\,(\mathsf{g}\,\top) \qquad \mathsf{a} \not\approx \mathsf{b}$$

The set is unsatisfiable because the instantiation $\{y \mapsto \lambda x.\,\mathsf{g}\,(x \mathrel{\boldsymbol{\approx}} \mathsf{a})\}$ produces the clause $\mathsf{h}\,(\mathsf{g}\,(\mathsf{b} \mathrel{\boldsymbol{\approx}} \mathsf{a})) \not\approx \mathsf{h}\,(\mathsf{g}\,\bot) \vee \mathsf{h}\,(\mathsf{g}\,(\mathsf{a} \mathrel{\boldsymbol{\approx}} \mathsf{a})) \not\approx \mathsf{h}\,(\mathsf{g}\,\top)$, which is unsatisfiable in conjunction with $\mathsf{a} \not\approx \mathsf{b}$.

The literal selection function can select either literal in the first clause. ERES is applicable in either case, but the unifiers $\{y \mapsto \lambda x.\,\mathsf{g}\,\bot\}$ and $\{y \mapsto \lambda x.\,\mathsf{g}\,\top\}$ do not lead to a contradiction. Instead, we need to apply FLUIDBOOLHOIST if the first literal is selected or FLUIDLOOBHOIST if the second literal is selected. In the first case, the derivation is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{a} \not\approx \mathsf{b} \qquad \dfrac{\dfrac{\dfrac{\mathsf{h}\,(y\,\mathsf{b}) \not\approx \mathsf{h}\,(\mathsf{g}\,\bot) \vee \mathsf{h}\,(y\,\mathsf{a}) \not\approx \mathsf{h}\,(\mathsf{g}\,\top)}{\mathsf{h}\,(z'\,\mathsf{b}\,\bot) \not\approx \mathsf{h}\,(\mathsf{g}\,\bot) \vee \mathsf{h}\,(z'\,\mathsf{a}\,(x'\mathsf{a})) \not\approx \mathsf{h}\,(\mathsf{g}\,\top) \vee x'\,\mathsf{b} \approx \top}\,\textsc{FluidBoolHoist}}{\mathsf{h}\,(\mathsf{g}\,(x'\,\mathsf{a})) \not\approx \mathsf{h}\,(\mathsf{g}\,\top) \vee x'\,\mathsf{b} \approx \top}\,\textsc{ERes}}{\mathsf{h}\,(\mathsf{g}\,(x''\,\mathsf{a} \mathrel{\boldsymbol{\approx}} x'''\,\mathsf{a})) \not\approx \mathsf{h}\,(\mathsf{g}\,\top) \vee \bot \approx \top \vee x''\,\mathsf{b} \approx x'''\,\mathsf{b}}\,\textsc{EqHoist}}{\mathsf{h}\,(\mathsf{g}\,(\mathsf{a} \mathrel{\boldsymbol{\approx}} x'''\,\mathsf{a})) \not\approx \mathsf{h}\,(\mathsf{g}\,\top) \vee \bot \approx \top \vee \mathsf{a} \not\approx x'''\,\mathsf{b}}\,\textsc{Sup}}{\mathsf{h}\,(\mathsf{g}\,\top) \not\approx \mathsf{h}\,(\mathsf{g}\,\top) \vee \bot \approx \top \vee \mathsf{a} \not\approx \mathsf{a}}\,\textsc{BoolRw}}{\dfrac{\dfrac{\bot \approx \top \vee \mathsf{a} \not\approx \mathsf{a}}{\bot \approx \top}\,\textsc{ERes}}{\bot}\,\textsc{FalseElim}}\,\textsc{ERes}$$

The FLUIDBOOLHOIST inference uses the unifier $\{y \mapsto \lambda u.\,z'\,u\,(x'\,u),\ z \mapsto \lambda u.\,z'\,\mathsf{b}\,u,\ x \mapsto x'\,\mathsf{b}\} \in \mathrm{CSU}(z\,x, y\,\mathsf{b})$. We apply ERES to the first literal of the resulting clause, with unifier $\{z' \mapsto \lambda uv.\,\mathsf{g}\,v\} \in \mathrm{CSU}(\mathsf{h}\,(z'\,\mathsf{b}\,\bot),\ \mathsf{h}\,(\mathsf{g}\,\bot))$. Next, we apply EQHOIST with the unifier $\{x' \mapsto \lambda u.\,x''\,u \mathrel{\boldsymbol{\approx}} x'''\,u,\ w \mapsto x''\,\mathsf{b},\ w' \mapsto x'''\,\mathsf{b}\} \in \mathrm{CSU}(x'\,\mathsf{b}, w \mathrel{\boldsymbol{\approx}} w')$ to the literal created by FLUIDBOOLHOIST, effectively performing a primitive substitution. The resulting clause can superpose into $\mathsf{a} \not\approx \mathsf{b}$ with the unifier $\{x'' \mapsto \lambda u.\,u\} \in \mathrm{CSU}(x''\,\mathsf{b}, \mathsf{b})$. The two sides of the interpreted equality in the first literal can then be unified, allowing us to apply BOOLRW with the unifier $\{y \mapsto \mathsf{a},\ x''' \mapsto \lambda u.\,\mathsf{a}\} \in \mathrm{CSU}(y \mathrel{\boldsymbol{\approx}} y, \mathsf{a} \mathrel{\boldsymbol{\approx}} x'''\,\mathsf{b})$. Finally, applying ERES twice and FALSEELIM once yields the empty clause.

Remarkably, none of the provers that participated in the CASC-J10 competition can solve this two-clause problem within a minute. Satallax finds a proof after 72 s and LEO-II after over 7 minutes. The CASC-28 version of our new Zipperposition implementation solves it in 3 s.

3.5 Soundness

All of our inference rules and axioms are sound w.r.t. $\models_{\approx}$ and the ones that do not introduce Skolem symbols are also sound w.r.t. $\models$. The preprocessing is sound w.r.t. both $\models$ and $\models_{\approx}$:

Lemma 28. *$\mathsf{Q}_{\approx}$-normalization preserves denotations of terms and truth of clauses w.r.t. proper interpretations.*

Proof. It suffices to show that

$$[\![\forall\langle\tau\rangle]\!]_{\mathfrak{I}}^{\xi} = [\![\lambda y.\, y \approx (\lambda x.\, \top)]\!]_{\mathfrak{I}}^{\xi} \quad \text{and} \quad [\![\exists\langle\tau\rangle]\!]_{\mathfrak{I}}^{\xi} = [\![\lambda y.\, y \not\approx (\lambda x.\, \bot)]\!]_{\mathfrak{I}}^{\xi}$$

for all types τ, proper interpretations $\mathfrak{I} = (\mathfrak{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$, and all valuations ξ.

Let f be a function from $[\![\tau]\!]_{\mathfrak{I}_{\mathsf{ty}}}^{\xi}$ to $\{0,1\}$. Then

$$[\![\forall\langle\tau\rangle]\!]_{\mathfrak{I}}^{\xi}(f) = \mathcal{J}(\forall, [\![\tau]\!]_{\mathfrak{I}_{\mathsf{ty}}}^{\xi})(f) = \min\{f(a) \mid a \in [\![\tau]\!]_{\mathfrak{I}_{\mathsf{ty}}}^{\xi}\} = \begin{cases} 1 & \text{if } f \text{ is constantly } 1 \\ 0 & \text{otherwise} \end{cases}$$

By the definition of proper interpretations, we have

$$[\![\lambda y.\, y \approx (\lambda x.\, \top)]\!]_{\mathfrak{I}}^{\xi}(f) = [\![y \approx (\lambda x.\, \top)]\!]_{\mathfrak{I}}^{\xi[y\mapsto f]} = \begin{cases} 1 & \text{if } [\![y]\!]_{\mathfrak{I}}^{\xi[y\mapsto f]} = [\![\lambda x.\, \top]\!]_{\mathfrak{I}}^{\xi[y\mapsto f]} \\ 0 & \text{otherwise} \end{cases}$$

Thus it remains to show that $[\![y]\!]_{\mathfrak{I}}^{\xi[y\mapsto f]} = [\![\lambda x.\, \top]\!]_{\mathfrak{I}}^{\xi[y\mapsto f]}$ if and only if f is constantly 1. This holds because by the definition of term denotation, $[\![y]\!]_{\mathfrak{I}}^{\xi[y\mapsto f]} = f$ and because $[\![\lambda x.\, \top]\!]_{\mathfrak{I}}^{\xi[y\mapsto f]}(a) = [\![\top]\!]_{\mathfrak{I}}^{\xi[x\mapsto a,\, y\mapsto f]} = 1$ by properness of the interpretation, for all $a \in [\![\tau]\!]_{\mathfrak{I}_{\mathsf{ty}}}^{\xi}$. The case of $\exists$ is analogous. $\square$

To show soundness of the inferences, we need the substitution lemma for our logic:

Lemma 29 (Substitution lemma). *Let $\mathfrak{I} = (\mathfrak{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$ be a proper interpretation. Then*

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

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

Proof. Analogous to Lemma 18 of λSup [7].

It follows that a model of a clause is also a model of its instances:

Lemma 30. *If $\mathfrak{I} \models C$ for some interpretation $\mathfrak{I}$ and some clause C, then $\mathfrak{I} \models C\rho$ for all substitutions ρ.*

Proof. Analogous to Lemma 19 of λSup [7], using Lemma 29.

With this lemma in place, we can prove the soundness of our calculus. Some of the rules and axioms are only sound w.r.t. $\models_{\approx}$.

Theorem 31 (Soundness). *Axiom* (CHOICE) *and all of our inference rules, except for* FORALLRW *and* EXISTSRW, *are sound w.r.t.* $\models$. *All of our axioms and inference rules are sound w.r.t.* $\models_{\approx}$. *Both of these claims hold even without the variable condition and the side conditions on fluidity, deeply occurring variables, order, and eligibility.*

Proof. Analogous to Lemma 20 of λSup [7]. For the Boolean rules, we make use of the special requirements on interpretations of logical symbols.

We elaborate on the soundness of FORALLRW, EXISTSRW, and EXT w.r.t. $\models_{\approx}$.

For FORALLRW: Let $\mathcal{I}$ be a Skolem-aware model of $C\langle u\rangle$. By Lemma 29, $\mathcal{I}$ is a model of $C\langle u\rangle\sigma$ as well. Since $\sigma \in \mathrm{CSU}(\forall\langle\beta\rangle\, y,\, u)$, we have $C\langle u\rangle\sigma = C\langle\forall\langle\beta\rangle\, y\rangle\sigma$. Thus, to show that $\mathcal{I}$ is also a model of the conclusion $C\langle y\,(\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\neg\, y\sigma\, z}\langle\bar{\alpha}\rangle\,\bar{x})\rangle\sigma$, it suffices to show that $\mathcal{I} \models \forall\langle\beta\sigma\rangle\,(y\sigma) \approx y\sigma\,(\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\neg\, y\sigma\, z}\langle\bar{\alpha}\rangle\,\bar{x})$. This follows directly from the definition of Skolem-awareness, which states that

$$\mathcal{I} \models (\exists\langle\beta\sigma\rangle\,(\lambda z.\,\neg\, y\sigma\, z)) \approx \neg\, y\sigma\,(\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,\neg\, y\sigma\, z}\langle\bar{\alpha}\rangle\,\bar{x})$$

For EXISTSRW, we can argue analogously.

For (EXT), we must show that any Skolem-aware model $\mathcal{I}$ is a model of axiom (EXT) $z\,(\mathsf{diff}\langle\alpha,\beta\rangle\, z\, y) \not\approx y\,(\mathsf{diff}\langle\alpha,\beta\rangle\, z\, y) \vee z \approx y$. By the definition of Skolem-awareness, we have $\mathcal{I} \models (\exists\langle\alpha\rangle\,(\lambda x.\, z\, x \not\approx y\, x)) \approx (\lambda x.\, z\, x \not\approx y\, x)\,(\mathsf{diff}\langle\alpha,\beta\rangle\, z\, y)$. Thus, if the first literal of (EXT) is false in $\mathcal{I}$ for some valuation ξ, then

$$\begin{aligned} 0 &= [\![(\lambda x.\, z\, x \not\approx y\, x)\,(\mathsf{diff}\langle\alpha,\beta\rangle\, z\, y)]\!]_{\mathcal{I}}^{\xi} \\ &= [\![\exists\langle\alpha\rangle\,(\lambda x.\, z\, x \not\approx y\, x)]\!]_{\mathcal{I}}^{\xi} \\ &= \max\{[\![\lambda x.\, z\, x \not\approx y\, x]\!]_{\mathcal{I}}^{\xi}(a) \mid a \in [\![\alpha]\!]_{\mathcal{I}}^{\xi}\} \\ &= \max\{[\![z\, x \not\approx y\, x]\!]_{\mathcal{I}}^{\xi[x\mapsto a]} \mid a \in [\![\alpha]\!]_{\mathcal{I}}^{\xi}\} \end{aligned}$$

It follows that there exists no $a \in [\![\alpha]\!]_{\mathcal{I}}^{\xi}$ such that $[\![z\, x]\!]_{\mathcal{I}}^{\xi[x\mapsto a]} = [\![z]\!]_{\mathcal{I}}^{\xi}(a)$ and $[\![y\, x]\!]_{\mathcal{I}}^{\xi[x\mapsto a]} = [\![y]\!]_{\mathcal{I}}^{\xi}(a)$ are different. Thus, $[\![z]\!]_{\mathcal{I}}^{\xi} = [\![y]\!]_{\mathcal{I}}^{\xi}$ and hence the second literal of (EXT) must be true under $\mathcal{I}$ and ξ. ☐

3.6 The Redundancy Criterion

As in λfSup and λSup, the redundancy criterion and the completeness proof distinguish three levels of logics. We have a higher-order level H, a $\mathsf{Q}_{\approx}$-normal ground higher-order level GH, and a ground monomorphic first-order level GF with an interpreted Boolean type. We use $\mathcal{T}_{\mathrm{H}}$, $\mathcal{T}_{\mathrm{GH}}$, and $\mathcal{T}_{\mathrm{GF}}$ to denote the respective sets of terms, $\mathcal{Ty}_{\mathrm{H}}$, $\mathcal{Ty}_{\mathrm{GH}}$, and $\mathcal{Ty}_{\mathrm{GF}}$ 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. We will define a grounding function $\mathcal{G}$ that connects levels H and GH and an encoding $\mathcal{F}$ that connects levels GH and GF.

Let $(\Sigma_{\mathsf{ty}}, \Sigma)$ be the signature of level H. The level GH has the same signature but is restricted to ground $\mathsf{Q}_{\approx}$-normal terms and clauses. For the GF level, we employ the logic of oSup [27]. Its signature $(\Sigma_{\mathrm{ty}}, \Sigma_{\mathrm{GF}})$ is defined as follows. The type constructors Σ_{ty} are the same in both signatures, but $\to$ is an uninterpreted type constructor on GF and not to be confused with the arrow used for type declarations in the logic of oSup [27], which we will avoid in this report due to the ambiguity.

For each ground instance $\mathsf{f}\langle\bar{\upsilon}\rangle : \tau_1 \to \cdots \to \tau_n \to \tau$ of a symbol $\mathsf{f} \in \Sigma$, we introduce a first-order symbol $\mathsf{f}_j^{\bar{\upsilon}} \in \Sigma_{\mathrm{GF}}$ with argument types $\bar{\tau}_j$ and result type $\tau_{j+1} \to \cdots \to \tau_n \to \tau$, for each j. This is done for both logical and nonlogical symbols. Moreover, for each ground term $\lambda x.\, t$, we introduce a symbol $\mathsf{lam}_{\lambda x.\, t} \in \Sigma_{\mathrm{GF}}$ of the same type. The symbols $\mathbf{\bot}_0$, $\mathbf{\top}_0$, $\mathbf{\neg}_1$, $\mathbf{\wedge}_2$, $\mathbf{\vee}_2$, $\mathbf{\rightarrow}_2$, $\mathbf{\approx}_2^\tau$, and $\mathbf{\not\approx}_2^\tau$ are identified with the corresponding first-order logical symbols.

Definition 32 (The grounding function $\mathcal{G}$ on terms and clauses). Given a clause $C \in \mathcal{C}_{\mathrm{H}}$, let its *ground instances* $\mathcal{G}(C)$ be the set of all clauses in $\mathcal{C}_{\mathrm{GH}}$ of the form $C\theta$ for some grounding substitution θ such that for all variables x occurring in C, the only Boolean green subterms of $x\theta$ are $\mathbf{\top}$ and $\mathbf{\bot}$.

Restricting the grounding to the Boolean terms $\mathbf{\top}$ and $\mathbf{\bot}$ allows condition O4 to consider only terms u with Boolean subterms $\mathbf{\top}$ and $\mathbf{\bot}$. This is crucial because without the restriction no suitable term order would exist. The approach resembles basic superposition [4], where the redundancy criterion only considers ground instances that are irreducible w.r.t. an arbitrary term rewriting system. A disadvantage of basic superposition is that its redundancy criterion severely restricts the simplification machinery because the irreducible terms are unknown during a derivation. In our setting, however, we know that $\mathbf{\top}$ and $\mathbf{\bot}$ will be normal forms of the term rewriting system used in the completeness proof. Thus we can restrict grounding to the Boolean terms $\mathbf{\top}$ and $\mathbf{\bot}$ without compromising the simplification machinery.

Since we have defined all clauses in $\mathcal{C}_{\mathrm{GH}}$ to be $\mathsf{Q}_{\approx}$-normal, the ground instances $\mathcal{G}(C)$ of a clause are $\mathsf{Q}_{\approx}$-normal as well. The clauses in $\mathcal{C}_{\mathrm{GH}}$ being $\mathsf{Q}_{\approx}$-normal allow us to define the encoding $\mathcal{F}$ as follows:

Definition 33 (The encoding $\mathcal{F}$ on terms and clauses). The encoding $\mathcal{F} : \mathcal{T}_{\mathrm{GH}} \to \mathcal{T}_{\mathrm{GF}}$ is recursively defined by

$$\mathcal{F}(\mathbf{\forall}\langle\tau\rangle\,(\lambda x.\, t)) = \forall x.\, \mathcal{F}(t) \qquad \mathcal{F}(\mathbf{\exists}\langle\tau\rangle\,(\lambda x.\, t)) = \exists x.\, \mathcal{F}(t)$$
$$\mathcal{F}(x) = x \qquad \mathcal{F}(\lambda x.\, t) = \mathsf{lam}_{\lambda x.\, t} \qquad \mathcal{F}(\mathsf{f}\langle\bar{\upsilon}\rangle\,\bar{s}_j) = \mathsf{f}_j^{\bar{\upsilon}}(\mathcal{F}(\bar{s}_j))$$

using $\downarrow_{\beta\eta\mathsf{Q}_\eta}$ representatives of terms. The encoding $\mathcal{F}$ is extended to map from $\mathcal{C}_{\mathrm{GH}}$ to $\mathcal{C}_{\mathrm{GF}}$ by mapping each literal and each side of a literal individually.

The encoding of variables is necessary for variables bound by $\mathbf{\forall}$ and $\mathbf{\exists}$. Since the terms $\mathcal{T}_{\mathrm{GH}}$ are $\mathsf{Q}_{\approx}$-normal, these variables occur neither applied nor inside λ-expressions.

Schematically, the three levels are connected as follows:

H		GH		GF
higher-order	$\xrightarrow{\quad\mathcal{G}\quad}$	ground $\mathsf{Q}_{\approx}$-normal higher-order	$\xrightarrow{\quad\mathcal{F}\quad}$	ground first-order with interpreted Booleans

The mapping $\mathcal{F}$ is clearly bijective. Using the inverse mapping, the order $\succ$ can be transferred from $\mathcal{T}_{\mathrm{GH}}$ to $\mathcal{T}_{\mathrm{GF}}$ and from $\mathcal{C}_{\mathrm{GH}}$ to $\mathcal{C}_{\mathrm{GF}}$ by defining $t \succ s$ as $\mathcal{F}^{-1}(t) \succ \mathcal{F}^{-1}(s)$ and $C \succ D$ as $\mathcal{F}^{-1}(C) \succ \mathcal{F}^{-1}(D)$. 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.

A key property of $\mathcal{F}$ is that there is a bijection between green subterms on GH and subterms on GF that are not below quantifiers:

Lemma 34. *Let $s, t \in \mathcal{T}_{\mathrm{GH}}$. If p is a green position in t or a position in $\mathcal{F}(t)$ that is not below a quantifier, 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 that is not below a quantifier.*

Proof. Analogous to Lemma 3.17 of λfSup [6]. □

Lemma 35. *The relation $\succ$ on $\mathcal{T}_{\mathrm{GF}}$ is a term order in the sense of* oSup. *That is, it is a well-founded strict order $\succ$ on ground terms such that*

1. *compatibility with contexts holds, but not necessarily below quantifiers;*
2. *the subterm property holds, but not necessarily below quantifiers;*
3. *totality holds;*
4. *$u \succ \bot_0 \succ \top_0$ for any term $u \in \mathcal{T}_{\mathrm{GF}}$ that is not $\top_0$ or $\bot_0$; and*
5. *$\mathsf{Q}x.\,t \succ t\{x \mapsto u\}$ for any term $u \in \mathcal{T}_{\mathrm{GF}}$ whose only Boolean subterms are $\top$ and $\bot$.*

Proof. Transitivity and irreflexivity follow directly from transitivity and irreflexivity of $\succ$ on $\mathcal{T}_{\mathrm{GH}}$. Well-foundedness, compatibility with contexts, subterm property and totality can be shown analogously to Lemma 3.19 of λfSup [6], using Lemma 34. That $\top$ or $\bot$ are the smallest terms follows from 3 of Definition 15. Finally, $\mathsf{Q}x.\,t \succ t\{x \mapsto u\}$ follows from 4 of Definition 15. □

Each of the three levels has an entailment relation $\models$. A clause set N_1 entails a clause set N_2, denoted by $N_1 \models N_2$, if all models of N_1 are also a models of N_2. For H and GH, we use higher-order models; for GF, we use first-order models with interpreted Booleans as defined by oSup [27]. We write $N_1 \models_{\mathcal{F}} N_2$ to abbreviate $\mathcal{F}(N_1) \models \mathcal{F}(N_2)$ and similarly for $N_1 \models_{\mathcal{G}} N_2$. On the H level, we additionally define Skolem-aware entailment, denoted by $N_1 \mathrel{|\!\!\approx} N_2$, to hold if all Skolem-aware models of a clause set $N_1 \subseteq \mathcal{C}_{\mathrm{H}}$ are also models of a clause set $N_2 \subseteq \mathcal{C}_{\mathrm{H}}$.

We define the sets of redundant clauses w.r.t. a 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 $D \in \mathcal{G}(C')$.

The tiebreaker $\sqsupset$ can be an arbitrary well-founded partial order on $\mathcal{C}_{\mathrm{H}}$, natural candidates being restrictions of (ill-founded) strict subsumption [7, Sect. 3.4].

Each of the three levels has an associated inference system *HInf*, *GHInf*, and *GFInf*. For H, it is the inference system *HInf* consisting of the rules described above. We view axioms (EXT) and (CHOICE) as premiseless inference rules EXT and CHOICE, respectively. We fix the selection functions *HLitSel* and *HBoolSel* globally.

The system *GHInf* is parameterized by selection functions and a witness function, which are defined as follows.

Definition 36 **(GH level selection functions).** A GH level literal selection *GHLitSel* maps each clause $C \in \mathcal{C}_{\mathrm{GH}}$ to a subset of its literals. A GH level Boolean subterm selection function *GHBoolSel* maps each clause $C \in \mathcal{C}_{\mathrm{GH}}$ to a subset of its green positions with Boolean subterms. We require these selection functions to have the property that for every $C \in \mathcal{C}_{\mathrm{GH}}$, there exists a $D \in \mathcal{C}_{\mathrm{H}}$ with $C \in \mathcal{G}(D)$ for which the selections $\mathit{HLitSel}(D)$, $\mathit{HBoolSel}(D)$ and the selections $\mathit{GHLitSel}(C)$, $\mathit{GHBoolSel}(C)$ correspond.

Definition 37 **(Witness function).** A witness function *GHWit* maps a clause $C \in \mathcal{C}_{\mathrm{GH}}$ and a green position of a quantifier-headed term in C to a term $\mathit{GHWit}(C,p) \in \mathcal{T}_{\mathrm{GH}}$ such that $\mathsf{Q}\langle\tau\rangle\, t \succ t\, \mathit{GHWit}(C,p)$ if $C|_p = \mathsf{Q}\langle\tau\rangle\, t$.

The witness function will be used to provide appropriate Skolem terms that witness the existence of terms fulfilling the given predicate.

In our completeness proof, the choice of the GH level selection and witness functions will depend on the saturated clause set in the limit of the derivation. Since we do not know this clause set during the derivation, we need to consider all possible parameters in our redundancy criterion:

Definition 38 **(Set of parameter triples Q).** Let Q be the set of parameters triples $(\mathit{GHLitSel}, \mathit{GHBoolSel}, \mathit{GHWit})$ where *GHLitSel* and *GHBoolSel* are GH level selection functions and *GHWit* is a witness function.

We write GHInf^q with $q = (\mathit{GHLitSel}, \mathit{GHBoolSel}, \mathit{GHWit}) \in Q$ to specify the inference system for a given set of parameters. The rules of GHInf^q include SUP, ERES, EFACT, BOOLHOIST, FALSEELIM, EQHOIST, NEQHOIST, and BOOLRW with the restriction that all references to $\succsim$ are replaced by $\succeq$.

In addition, GHInf^q contains the rules GFORALLHOIST, GEXISTSHOIST, GARGCONG, GEXT, and GCHOICE, which enumerate ground terms in the conclusion where their *HInf* counterparts use fresh variables. They enumerate all terms $u \in \mathcal{T}_{\mathrm{GH}}$ such that the only Boolean green subterms of u are $\top$ and $\bot$. Let $\mathcal{T}^{\star}_{\mathrm{GH}}$ be the set of all such terms u. Then these rules are stated as follows:

$$\frac{C\langle \forall\langle\tau\rangle\, v\rangle_p}{C\langle\bot\rangle \vee v\, u \approx \top}\,\text{GFORALLHOIST} \qquad \frac{C\langle \exists\langle\tau\rangle\, v\rangle_p}{C\langle\top\rangle \vee v\, u \approx \bot}\,\text{GEXISTSHOIST}$$

where p is $\succeq$-eligible in C and $u \in \mathcal{T}^{\star}_{\mathrm{GH}}$.

$$\frac{C' \vee t \approx s}{C' \vee t\,\bar{u} \approx s\,\bar{u}}\,\text{GARGCONG}$$

where $t \approx s$ is strictly $\succeq$-eligible in $C' \vee t \approx s$ and $u_i \in \mathcal{T}^{\star}_{\mathrm{GH}}$.

The rules GEXT and GCHOICE are premise-free and their conclusions are the infinitely many $\mathcal{G}$-instances of (EXT) and (CHOICE), respectively.

Moreover, $GHInf^q$ contains the following two rules, which use the witness function *GHWit* instead of Skolem terms:

$$\frac{C\langle\forall\langle\tau\rangle\, v\rangle_p}{C\langle v\, GHWit(C,p)\rangle_p}\,\text{GForallRw} \qquad \frac{C\langle\exists\langle\tau\rangle\, v\rangle_p}{C\langle v\, GHWit(C,p)\rangle_p}\,\text{GExistsRw}$$

where p is $\succeq$-eligible in C, and, for GForallRw (resp. for GExistsRw), $\mathcal{F}(C\langle\top\rangle_p)$ (resp. $\mathcal{F}(C\langle\bot\rangle_p)$) is not a tautology.

The inference systems $GHInf^q$ are indeed inference systems on $\mathcal{C}_{\mathrm{GH}}$—i.e., if the premises are in $\mathcal{C}_{\mathrm{GH}}$, the conclusions are in $\mathcal{C}_{\mathrm{GH}}$, too. The conclusions are obviously ground. They are also $\mathsf{Q}_{\approx}$-normal:

Lemma 39. *If the premises of a $GHInf^q$ inference are $\mathsf{Q}_{\approx}$-normal, then the conclusion is also $\mathsf{Q}_{\approx}$-normal.*

Proof. The conclusions of GExt and GChoice are $\mathsf{Q}_{\approx}$-normal by the definition of $\mathcal{G}$.

The definition of $\mathsf{Q}_{\approx}$-normality clearly only depends on the contained quantifier-headed subterms. As long as no new quantifier-headed subterms are added, a clause set cannot become $\mathsf{Q}_{\approx}$-reducible.

The inference rules ERes, EFact, and FalseElim do not introduce any subterms that were not already present in the premises. The inference rules Sup, BoolHoist, EqHoist, NeqHoist, BoolRw only introduce new subterms by replacing a green subterm of a $\mathsf{Q}_{\approx}$-normal term by another $\mathsf{Q}_{\approx}$-normal term. Since green positions are never below quantifiers, these rules also do not add new quantifier-headed subterms.

For the inference rules GForallHoist, GExistsHoist, GArgCong, GForallRw, and GExistsRw, we can use Lemma 14 to show that the conclusions are $\mathsf{Q}_{\approx}$-normal. □

The system *GFInf* is parameterized by an analogous triple (*GFLitSel*, *GFBoolSel*, *GFWit*). Using the bijection $\mathcal{F}$, we can translate a parameter triple q of *GHInf* to a parameter triple $\mathcal{F}(q)$ of *GFInf*. Let $GFInf^{\mathcal{F}(q)}$ be the inference system containing the inferences isomorphic to $GHInf^q$ obtained by $\mathcal{F}$, except for GArgCong, GExt, and GChoice. This is essentially identical to the ground inference system of oSup [27].

We extend the functions $\mathcal{F}$ and $\mathcal{G}$ to inferences:

Notation 40. Given an inference ι, we write $prems(\iota)$ for the tuple of premises, $mprem(\iota)$ for the main (i.e., rightmost) premise, and $concl(\iota)$ for the conclusion.

Definition 41 (The encoding $\mathcal{F}$ on inferences). Given an inference $\iota \in GHInf$ that is not a GArgCong, GExt, or GChoice inference, let $\mathcal{F}(\iota) \in GFInf$ denote the inference defined by $prems(\mathcal{F}(\iota)) = \mathcal{F}(prems(\iota))$ and $concl(\mathcal{F}(\iota)) = \mathcal{F}(concl(\iota))$.

Definition 42 (The grounding function $\mathcal{G}$ on inferences). Given a parameter triple $q \in Q$ and an inference $\iota \in HInf$, we define the set $\mathcal{G}^q(\iota)$ of ground instances of ι to be all inferences $\iota' \in GHInf^q$ such that $prems(\iota') = prems(\iota)\theta$ and $concl(\iota') = concl(\iota)\theta$ for some grounding substitution θ.

Thus, $\mathcal{G}$ maps FLUIDSUP to SUP, FLUIDBOOLHOIST to BOOLHOIST, FORALLRW to GFORALLRW, EXISTSRW to GEXISTSRW, FORALLHOIST to GFORALLHOIST, EXISTSHOIST to GEXISTSHOIST, ARGCONG to GARGCONG, EXT to GEXT, CHOICE to GCHOICE, and inferences of other *HInf* rules to inferences of the identically named rules in *GHInf*. For FLUIDLOOBHOIST, which needs not be grounded to prove refutational completeness, we let $\mathcal{G}^q(\iota) = \mathit{undef}$.

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

- Given $\iota \in \mathit{GFInf}^q$ and $N \subseteq \mathcal{C}_{\mathrm{GF}}$, let $\iota \in \mathit{GFRed}_{\mathrm{I}}^q(N)$ 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 $\iota \in \mathit{GHInf}^q$ and $N \subseteq \mathcal{C}_{\mathrm{GH}}$, let $\iota \in \mathit{GHRed}_{\mathrm{I}}^q(N)$ if
  - ι is GARGCONG, GEXT, or GCHOICE and $\mathit{concl}(\iota) \in N \cup \mathit{GHRed}_{\mathrm{C}}(N)$; or
  - ι is any another inference and $\mathcal{F}(\iota) \in \mathit{GFRed}_{\mathrm{I}}^{\mathcal{F}(q)}(\mathcal{F}(N))$.
- Given $\iota \in \mathit{HInf}$ and $N \subseteq \mathcal{C}_{\mathrm{H}}$, let $\iota \in \mathit{HRed}_{\mathrm{I}}(N)$ if
  - ι is a FLUIDLOOBHOIST inference and $\mathcal{G}(\mathit{concl}(\iota)) \subseteq \mathcal{G}(N) \cup \mathit{GHRed}_{\mathrm{C}}(\mathcal{G}(N))$ or
  - ι is any other inference and $\mathcal{G}^q(\iota) \subseteq \mathit{GHRed}_{\mathrm{I}}(\mathcal{G}(N))$ for all $q \in Q$.

Some authors prefer not to define inferences with a redundant premise as redundant, but in our proof of refutational completeness, this will be crucial for the lifting lemma of FORALLRW and EXISTSRW.

A clause set N is *saturated* w.r.t. an inference system and the inference component $\mathit{Red}_{\mathrm{I}}$ of a redundancy criterion if every inference from clauses in N is in $\mathit{Red}_{\mathrm{I}}(N)$.

3.7 Simplification Rules

The redundancy criterion $(\mathit{HRed}_{\mathrm{I}}, \mathit{HRed}_{\mathrm{C}})$ is strong enough to support most simplification rules implemented in Schulz's first-order prover E [29, Sections 2.3.1 and 2.3.2], although some require minor adaptions. To describe the adaptions, we introduce the notion of blue subterms, which include all green subterms but also some subterms below quantifiers.

Definition 43 (Blue subterms and positions). Blue subterms and positions are inductively defined as follows. A blue position is a tuple of natural numbers. For any λ-term t, the empty tuple ε is a blue position of t, and t is the blue subterm of t at position ε. For all symbols $\mathsf{f} \in \Sigma \setminus \{\forall, \exists\}$, if t is a blue subterm of u_i at position p, then $i.p$ is a blue position of $\mathsf{f}\langle\bar{\tau}\rangle\,\bar{u}$, and t is the blue subterm of $\mathsf{f}\langle\bar{\tau}\rangle\,\bar{u}$ at position $i.p$. If t is a blue subterm of u at position p, then $1.p$ is a blue position of $\mathsf{Q}\langle\tau\rangle\,(\lambda x.\,u)$ and t is the blue subterm of $\mathsf{Q}\langle\tau\rangle\,(\lambda x.\,u)$ at position $1.p$.

For example, the blue subterms of $\mathsf{f}\;(\mathsf{g}\;(\neg\,\mathsf{p}))\;(\forall\langle\tau\rangle\;(\lambda x.\,\mathsf{q}))\;(y\;\mathsf{a})\;(\lambda x.\,\mathsf{h}\;\mathsf{b})$ are all of the green subterms and q. The notions of blue positions and subterms are lifted to $\beta\eta$-equivalence classes via the $\beta\eta\mathsf{Q}_\eta$-normal representative.

Deletion of duplicated literals, deletion of resolved literals, syntactic tautology deletion, negative simplify-reflect, and clause subsumption adhere to our redundancy criterion. Positive simplify-reflect and equality subsumption are supported by our criterion if they are applied on blue subterms. Semantic tautology deletion can be applied as well, but we must use the entailment relation $\models_{\mathcal{F}}$. Rewriting of positive and negative literals (demodulation) can only be applied on blue subterms. Moreover, for positive literals, the rewriting clause must be smaller than the rewritten clause—a condition that is also necessary with the standard first-order redundancy criterion but not always fulfilled by Schulz's rule. As for destructive equality resolution, even in first-order logic the rule cannot be justified with the standard redundancy criterion, and it is unclear whether it preserves refutational completeness.

As a representative example, we show how demodulation into green contexts can be justified. Demodulation into blue contexts and the other simplification rules can be justified similarly.

Lemma 44. *Demodulation into green contexts is a simplification:*

$$\dfrac{t \approx t' \quad \overbrace{s\langle t\sigma\rangle \,\dot{\approx}\, s' \vee C'}^{C}}{t \approx t' \quad s\langle t'\sigma\rangle \,\dot{\approx}\, s' \vee C'}\;\textsc{Demod}$$

where $t\sigma \succ t'\sigma$ and $C \succ (t \approx t')\sigma$. It adheres to our redundancy criterion—i.e., the deleted premise C is redundant w.r.t. the conclusions.

Proof. Let N be the set consisting of the two conclusions. We must show that $C \in \mathit{HRed}_{\mathrm{C}}(N)$. Let $C\theta$ be a ground instance of C. By definition of $\mathit{HRed}_{\mathrm{C}}$, it suffices to show that $C\theta \in \mathit{GHRed}_{\mathrm{C}}(\mathcal{G}(N))$. By definition of $\mathit{GHRed}_{\mathrm{C}}$, we must thus show that $\mathcal{F}(C\theta) \in \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(\mathcal{G}(N)))$. By definition of $\mathit{GFRed}_{\mathrm{C}}$, this is equivalent to proving that the clauses in $\mathcal{F}(\mathcal{G}(N))$ that are smaller than $\mathcal{F}(C\theta)$ entail $\mathcal{F}(C\theta)$.

By compatibility with green contexts and stability under substitutions of $\succ$, the condition $t\sigma \succ t'\sigma$ implies that $D = \mathcal{F}((s\langle t'\sigma\rangle \,\dot{\approx}\, s' \vee C')\theta)$ is a clause in $\mathcal{F}(\mathcal{G}(N))$ that is smaller than $\mathcal{F}(C\theta)$. By stability under substitutions, $C \succ (t \approx t')\sigma$ implies that $E = \mathcal{F}((t \approx t')\sigma\theta)$ is another clause in $\mathcal{F}(\mathcal{G}(N))$ that is smaller than $\mathcal{F}(C\theta)$. By Lemma 34, green subterms on the GH level correspond to subterms on the GF level. Thus, $\{D, E\} \models \mathcal{F}(C\theta)$ by congruence.

Under some circumstances, certain inference rules of our calculus can be applied as simplifications—i.e., a premise can be deleted after performing them. The FALSEELIM and BOOLRW rules can be applied as a simplification if σ is the identity. If the head of u is $\forall$, FORALLHOIST and FORALLRW can both be applied and, together, serve as one simplification rule. If the head of u is $\forall$ and FORALLRW cannot be applied due to its condition 5, FORALLHOIST alone serves as a simplification rule. The same holds for EXISTSHOIST and EXISTSRW if the head of u is $\exists$. For all of these simplifications, the eligibility conditions can be ignored.

If σ is the identity, the rule BOOLHOIST can also be applied as a simplification in combination with the following rule to the same subterm u:

$$\dfrac{C\langle u\rangle}{C\langle\top\rangle \vee u \approx \bot}\;\textsc{LoobHoist}$$

Again, the eligibility condition can be ignored, and u can even be a fully applied logical symbol as long as it is not $\top$ or $\bot$.

3.8 Clausification

Like oSup, our calculus does not require the input problem to be clausified during the preprocessing, and it supports higher-order analogues of the three inprocessing clausification methods introduced by Nummelin et al. *Inner delayed clausification* relies on our core calculus rules to destruct logical symbols. *Outer delayed clausification* adds the following clausification rules to the calculus:

$$\frac{s \approx \top \vee C}{oc(s, C)}\ \textsc{PosOuterClaus} \qquad \frac{s \approx \bot \vee C}{oc(\neg s, C)}\ \textsc{NegOuterClaus}$$

$$\frac{s \approx t \vee C}{s \approx \bot \vee t \approx \top \vee C \quad s \approx \top \vee t \approx \bot \vee C}\ \textsc{EqOuterClaus}$$

$$\frac{s \not\approx t \vee C}{s \approx \bot \vee t \approx \bot \vee C \quad s \approx \top \vee t \approx \top \vee C}\ \textsc{NeqOuterClaus}$$

The double bars identify simplification rules (i.e., the conclusions make the premise redundant and can replace it). The first two rules require that s has a logical symbol as its head, whereas the last two require that s and t are Boolean terms other than $\top$ and $\bot$. The function oc distributes the logical symbols over the clause C—e.g., $oc(s \rightarrow t,\, C) = \{s \approx \bot \vee t \approx \top \vee C\}$, and $oc(\neg(s \vee t),\, C) = \{s \approx \bot \vee C, t \approx \bot \vee C\}$. It is easy to check that our redundancy criterion allows us to replace the premise of the OuterClaus rules with their conclusion. Nonetheless, we apply EqOuterClaus and NeqOuterClaus as inferences because the premises might be useful in their original form.

Besides the two delayed clausification methods, a third inprocessing clausification method is *immediate* clausification. This clausifies the input problem's outer Boolean structure in one swoop, resulting in a set of higher-order clauses. If unclausified Boolean terms rise to the top during saturation, the same algorithm is run to clausify them.

Unlike delayed clausification, immediate clausification is a black box and is unaware of the proof state other than the Boolean term it is applied to. Delayed clausification, on the other hand, clausifies the term step by step, allowing us to interleave clausification with the strong simplification machinery of superposition provers. It is especially powerful in higher-order contexts: Examples such as $y\,\mathsf{p}\,\mathsf{q} \not\approx (\mathsf{p} \vee \mathsf{q})$ can be refuted directly by equality resolution, rather than via more explosive rules on the clausified form.

3.9 A Concrete Term Order

We define a concrete order $\succ_\lambda$ that fulfills the properties of a strict term order as defined in Definition 17 to show that the requirements can indeed be fulfilled and to provide a concrete order for implementations of our calculus.

Given a signature $(\Sigma_{\mathsf{ty}}, \Sigma)$, we encode types and terms as terms over the untyped first-order signature $\Sigma_{\mathsf{ty}} \uplus \{\mathsf{f}_k \mid \mathsf{f} \in \Sigma, k \in \mathbb{N}\} \uplus \{\mathsf{lam}, \forall'_1, \exists'_1\} \uplus \{\mathsf{db}^i_k \mid i, k \in \mathbb{N}\}$. We define the encoding in two parts. The first part is the encoding O, resembling the one defined for λSup. The auxiliary function $\mathcal{B}_x(t)$ replaces each free occurrence of the variable x

by a De Bruijn index—that is, a symbol db^i where i is the number of λ-expressions surrounding the variable occurrence. The encoding $\mathcal{O}$ recursively encodes higher-order types into untyped first-order terms as follows: $\mathcal{O}(\alpha) = \alpha$ and $\mathcal{O}(\kappa(\bar{\tau})) = \kappa(\mathcal{O}(\bar{\tau}))$. Using $\beta\eta\mathsf{Q}_\eta$-normal representatives, it recursively encodes higher-order terms into untyped first-order terms as follows:

$$\mathcal{O}(t) = \begin{cases} z_t & \text{if } t = x \text{ or } t \text{ is fluid} \\ \mathsf{lam}(\mathcal{O}(\tau), \mathcal{O}(\mathcal{B}_x(u))) & \text{if } t = (\lambda x : \tau.\, u) \text{ and } t \text{ is not fluid} \\ \mathsf{f}_k(\mathcal{O}(\bar{\tau}), \mathcal{O}(\bar{u}_k)) & \text{if } t = \mathsf{f}\langle\bar{\tau}\rangle\, \bar{u}_k \text{ and either } \mathsf{f} \notin \{\forall, \exists\} \text{ or } k = 0 \\ \mathsf{Q}_1(\mathcal{O}(\tau), \mathcal{O}(\mathcal{B}_x(u))) & \text{if } t = \mathsf{Q}\langle\tau\rangle(\lambda x : \tau.\, u) \end{cases}$$

Via this encoding, the term order conditions (O1), (O2), and (O3) can be easily achieved. For (O4), however, we need to transform the encoded term further to ensure that the symbols $\forall_1$ and $\exists_1$ occur only as translations of fully applied quantifiers in green contexts. Then we can achieve (O4) by assigning them a large weight. The function $\mathcal{P}$ transforms the result of $\mathcal{O}$ in this way by applying a function p to all subterms below lam symbols.

$$\mathcal{P}(t) = \begin{cases} \alpha & \text{if } t = \alpha \\ z_u & \text{if } t = z_u \\ \mathsf{lam}(\mathcal{P}(\tau), p(u)) & \text{if } t = \mathsf{lam}(\tau, u) \\ \mathsf{f}(\mathcal{P}(\bar{t})) & \text{if } t = \mathsf{f}(\bar{t}) \text{ and } \mathsf{f} \neq \mathsf{lam} \end{cases}$$

The function p replaces $\forall_1$ by $\forall'_1$, $\exists_1$ by $\exists'_1$, and z_u by a fresh variable z'_u.

$$p(t) = \begin{cases} \alpha & \text{if } t = \alpha \\ z'_u & \text{if } t = z_u \\ \forall'_1(p(\tau), p(u)) & \text{if } t = \forall_1(\tau, u) \\ \exists'_1(p(\tau), p(u)) & \text{if } t = \exists_1(\tau, u) \\ \mathsf{f}(p(\bar{t})) & \text{if } t = \mathsf{f}(\bar{t}) \text{ and } \mathsf{f} \notin \{\forall_1, \exists_1\} \end{cases}$$

For example, $\mathcal{O}$ encodes the term $\forall\langle\iota\rangle(\lambda x.\, \mathsf{p}\, y\, y\, (\lambda u.\, \mathsf{f}\, y\, y\, (\forall\langle\iota\rangle(\lambda v.\, u))))$ into the first-order term $\forall_1(\iota, \mathsf{p}_3(z_y, z_y, \mathsf{lam}(\mathsf{o}, \mathsf{f}_3(z_y, z_y, \forall_1(\iota, \mathsf{db}^1)))))$ and $\mathcal{P}$ transforms it into $\forall_1(\iota, \mathsf{p}_3(z_y, z_y, \mathsf{lam}(\mathsf{o}, \mathsf{f}_3(z'_y, z'_y, \forall'_1(\iota, \mathsf{db}^1)))))$.

Using the encoding $\mathcal{O}$ and the function $\mathcal{P}$, we define our term order $\succ_\lambda$. Let $\succ_{\mathsf{kb}}$ be the transfinite Knuth–Bendix order [25] on first-order terms. The weight of $\forall_1$ and $\exists_1$ must be ω, the weight of $\top_0$ and $\bot_0$ must be 1, and the weights of all other symbols must be smaller than ω. The precedence $>$ must be total and $\bot_0 > \top_0$ must be the symbols of lowest precedence. We do not use subterm coefficients (i.e., all coefficients are 1), nor a symbol of weight 0. The precedence function is arbitrary. Let $\succ_{\mathcal{P}}$ be the order induced by $\mathcal{P}$ from $\succ_{\mathsf{kb}}$, meaning $t \succ_{\mathcal{P}} s$ if and only if $\mathcal{P}(t) \succ_{\mathsf{kb}} \mathcal{P}(s)$. Let $\succ_\lambda$ be the order induced by $\mathcal{O}$ from $\succ_{\mathcal{P}}$, meaning $t \succ_\lambda s$ if and only if $\mathcal{O}(t) \succ_{\mathcal{P}} \mathcal{O}(s)$. We extend $\succ_\lambda$ to literals and clauses in the usual way. We will show that $\succ_\lambda$ fulfills the properties of a strict term order:

Lemma 45. *The restriction of $\succ_\lambda$ to ground terms is a strict ground term order, as defined in Definition 15.*

Proof. We follow the proof of Lemma 31 of Bentkamp et al.

The transfinite Knuth–Bendix order $\succ_{\mathsf{kb}}$ has been shown to enjoy irreflexivity, transitivity, well-foundedness, totality on ground terms, the subterm property, and compatibility with contexts [25].

Transitivity and irreflexivity of $\succ_{\mathsf{kb}}$ imply transitivity and irreflexivity of $\succ_\lambda$, respectively.

WELL-FOUNDEDNESS: If there existed an infinite chain $t_1 \succ_\lambda t_2 \succ_\lambda \cdots$ of ground terms, there would also be the chain $\mathcal{P}(\mathcal{O}(t_1)) \succ_{\mathsf{kb}} \mathcal{P}(\mathcal{O}(t_2)) \succ_{\mathsf{kb}} \cdots$, contradicting the well-foundedness of $\succ_{\mathsf{kb}}$.

TOTALITY: For any ground terms t and s we have $\mathcal{P}(\mathcal{O}(t)) \succ_{\mathsf{kb}} \mathcal{P}(\mathcal{O}(s))$, $\mathcal{P}(\mathcal{O}(t)) \prec_{\mathsf{kb}} \mathcal{P}(\mathcal{O}(s))$, or $\mathcal{P}(\mathcal{O}(t)) = \mathcal{P}(\mathcal{O}(s))$ by ground totality of $\succ_{\mathsf{kb}}$. In the first two cases, it follows that $t \succ_\lambda s$ or $t \prec_\lambda s$ respectively. In the last case, it follows that $t = s$ because $\mathcal{O}$ and $\mathcal{P}$ are clearly injective.

(O1): By induction on the depth of the context, it suffices to show that $t \succ_\lambda s$ implies $\mathsf{f}\langle\bar\tau\rangle\,\bar u\,t\,\bar v \succ_\lambda \mathsf{f}\langle\bar\tau\rangle\,\bar u\,s\,\bar v$ for all t, s, $\mathsf{f} \in \Sigma \setminus \{\forall, \exists\}$, $\bar\tau$, $\bar u$, and $\bar v$. This amounts to showing that $\mathcal{P}(\mathcal{O}(t)) \succ_{\mathsf{kb}} \mathcal{P}(\mathcal{O}(s))$ implies

$$\begin{aligned}
&\mathcal{P}(\mathcal{O}(\mathsf{f}\langle\bar\tau\rangle\,\bar u\,t\,\bar v)) = \mathcal{P}(\mathsf{f}_k(\mathcal{O}(\bar\tau), \mathcal{O}(\bar u), \mathcal{O}(t), \mathcal{O}(\bar v))) = \\
&\mathsf{f}_k(\mathcal{P}(\mathcal{O}(\bar\tau)), \mathcal{P}(\mathcal{O}(\bar u)), \mathcal{P}(\mathcal{O}(t)), \mathcal{P}(\mathcal{O}(\bar v))) \succ_{\mathsf{kb}} \mathsf{f}_k(\mathcal{P}(\mathcal{O}(\bar\tau)), \mathcal{P}(\mathcal{O}(\bar u)), \mathcal{P}(\mathcal{O}(s)), \mathcal{P}(\mathcal{O}(\bar v))) \\
&= \mathcal{P}(\mathsf{f}_k(\mathcal{O}(\bar\tau), \mathcal{O}(\bar u), \mathcal{O}(s), \mathcal{O}(\bar v))) = \mathcal{P}(\mathcal{O}(\mathsf{f}\langle\bar\tau\rangle\,\bar u\,s\,\bar v))
\end{aligned}$$

which follows directly from compatibility of $\succ_{\mathsf{kb}}$ with contexts and the induction hypothesis.

(O2): Let s be a term. We show that $s \succeq_\lambda s|_p$ by induction on p, where $s|_p$ denotes the green subterm at position p. If $p = \varepsilon$, this is trivial. If $p = p'.i$, we have $s \succeq_\lambda s|_{p'}$ by the induction hypothesis. Hence, it suffices to show that $s|_{p'} \succeq_\lambda s|_{p'.i}$. From the existence of the position $p'.i$, we know that $s|_{p'}$ must be of the form $s|_{p'} = \mathsf{f}\langle\bar\tau\rangle\,\bar u_k$ for some $\mathsf{f} \in \Sigma \setminus \{\forall, \exists\}$. Then $s|_{p'.i} = u_i$. The encoding yields $\mathcal{P}(\mathcal{O}(s|_{p'})) = \mathcal{P}(\mathsf{f}_k(\mathcal{O}(\bar\tau), \mathcal{O}(\bar u_k))) = \mathsf{f}_k(\mathcal{P}(\mathcal{O}(\bar\tau)), \mathcal{P}(\mathcal{O}(\bar u_k)))$ and hence $\mathcal{P}(\mathcal{O}(s|_{p'})) \succeq_{\mathsf{kb}} \mathcal{P}(\mathcal{O}(s|_{p'.i}))$ by the subterm property of $\succ_{\mathsf{kb}}$. Hence, $s|_{p'} \succeq_\lambda s|_{p'.i}$ and thus $s \succeq_\lambda s|_p$.

(O3): Since we do not have a symbol of weight 0, and $\top_0$ and $\bot_0$ have weight 1, there cannot be any term of smaller weight. Since moreover $\top_0$ and $\bot_0$ have the lowest precedence, they are the smallest terms w.r.t. $\succ_{\mathsf{kb}}$. We have $\bot_0 \succ_{\mathsf{kb}} \top_0$ because $\bot_0$ has higher precedence. Since $\mathcal{P}(\mathcal{O}(\top)) = \top_0$ and $\mathcal{P}(\mathcal{O}(\top)) = \bot_0$, it follows that $\top$ and $\bot$ are the smallest ground terms w.r.t. $\succ_\lambda$ and that $\bot \succ_\lambda \top$.

(O4): Let $\mathsf{Q}\langle\tau\rangle\,t$ and u be $\mathsf{Q}_\approx$-normal ground terms. We assume that the Boolean green subterms of u are $\top$ and $\bot$. We must show $\mathsf{Q}\langle\tau\rangle\,t \succ_\lambda t\,u$, which is equivalent to $\mathcal{P}(\mathcal{O}(\mathsf{Q}\langle\tau\rangle\,t)) \succ_{\mathsf{kb}} \mathcal{P}(\mathcal{O}(t\,u))$.

All symbols except $\forall_1$ and $\exists_1$ have finite weight. Only $\forall_1$ and $\exists_1$ have weight ω. Since all subterm coefficients are 1, the coefficient of ω in the weight of a given term indicates the number of occurrences of the symbols $\forall_1$ and $\exists_1$ in that term.

In η-expanded form, we have $t = \lambda x.\, s$ for some s. Then we have $\mathcal{P}(O(\mathsf{Q}\langle\tau\rangle\, t)) = \mathsf{Q}_1(\mathcal{P}(O(\tau)), \mathcal{P}(O(\mathcal{B}_x(s)))))$ and $\mathcal{P}(O(t\,u)) = \mathcal{P}(O(s\{x \mapsto u\}))$. By $\mathsf{Q}_{\approx}$-normality, x occurs only in green positions of s. Therefore, replacing x by u in s does not trigger any $\beta\eta\mathsf{Q}_\eta$-normalizations. Thus, $\mathcal{P}(O(\mathcal{B}_x(s))))$ and $\mathcal{P}(O(s\{x \mapsto u\}))$ are almost identical, except that $\mathcal{P}(O(\mathcal{B}_x(s))))$ contains db^i where $\mathcal{P}(O(s\{x \mapsto u\}))$ contains $\mathcal{P}(O(u))$. Since the only Boolean green subterms of u are $\top$ and $\bot$, $\mathcal{P}(O(u))$ does not contain $\forall_1$ or $\exists_1$. So $\mathcal{P}(O(\mathcal{B}_x(s))))$ and $\mathcal{P}(O(s\{x \mapsto u\}))$ contain the same number of $\forall_1$ and $\exists_1$ symbols. Hence, $\mathcal{P}(O(\mathsf{Q}\langle\tau\rangle\, t))$ contains exactly one more of these symbols than $\mathcal{P}(O(t\,u))$. This means that the weight of the former is larger than the weight of the latter and thus $\mathcal{P}(O(\mathsf{Q}\langle\tau\rangle\, t)) \succ_{\mathsf{kb}} \mathcal{P}(O(t\,u))$. $\square$

Lemma 46. *The relation $\succ_\lambda$ is a strict term order as defined in Definition 17.*

Proof. Given Lemma 45, it remains to show that $\succ_\lambda$ is stable under grounding substitutions. Assume $s \succ_\lambda s'$ for some terms s and s'. Let θ be a higher-order substitution grounding s and s'. We must show $s\theta \succ_\lambda s'\theta$. We will define a first-order substitution ρ grounding $\mathcal{P}(O(s))$ and $\mathcal{P}(O(s'))$ such that $\mathcal{P}(O(s))\rho = \mathcal{P}(O(s\theta))$ and $\mathcal{P}(O(s'))\rho = \mathcal{P}(O(s'\theta))$. Since $s \succ_\lambda s'$, we have $\mathcal{P}(O(s)) \succ_{\mathsf{kb}} \mathcal{P}(O(s'))$. The transfinite Knuth–Bendix order $\succ_{\mathsf{kb}}$ has been shown to be stable under substitutions [25]. Hence, $\mathcal{P}(O(s))\rho \succ_{\mathsf{kb}} \mathcal{P}(O(s'))\rho$ and therefore $\mathcal{P}(O(s\theta)) \succ_{\mathsf{kb}} \mathcal{P}(O(s'\theta))$ and $s\theta \succ_\lambda s'\theta$.

We define the first-order substitution ρ as $\alpha\rho = \alpha\theta$ for type variables α, $z_u\rho = \mathcal{P}(O(u\theta))$, and $z'_u\rho = p(O(u\theta))$ for terms u. Strictly speaking, the domain of a substitution must be finite, so we restrict this definition of ρ to the finitely many variables that occur in the computation of $\mathcal{P}(O(s))$ and $\mathcal{P}(O(s'))$.

Clearly, we have $\mathcal{P}(O(\tau))\rho = \mathcal{P}(O(\tau\theta))$ and $p(O(\tau))\rho = p(O(\tau\theta))$ for all types τ occurring in the computation of $\mathcal{P}(O(s))$ and $\mathcal{P}(O(s'))$. Moreover, $\mathcal{P}(O(t))\rho = \mathcal{P}(O(t\theta))$ and $p(O(t))\rho = p(O(t\theta))$ for all terms t occurring in the computation of $\mathcal{P}(O(s))$ and $\mathcal{P}(O(s'))$, which we show by induction on the structure of t.

If $t = x$ or if t is fluid, $\mathcal{P}(O(t))\rho = z_t\rho = \mathcal{P}(O(t\theta))$ and $p(O(t))\rho = z'_t\rho = p(O(t\theta))$.

If $t = \mathsf{f}\langle\bar{\tau}\rangle\, \bar{u}$ for $\mathsf{f} \notin \{\forall, \exists\}$, then

$$\begin{aligned}\mathcal{P}(O(t))\rho &= \mathsf{f}_k(\mathcal{P}(O(\bar{\tau}))\rho, \mathcal{P}(O(\bar{u}))\rho)\\ &\stackrel{\text{IH}}{=} \mathsf{f}_k(\mathcal{P}(O(\bar{\tau}\theta)), \mathcal{P}(O(\bar{u}\theta))) = \mathcal{P}(O(\mathsf{f}\langle\bar{\tau}\theta\rangle\,(\bar{u}\theta))) = \mathcal{P}(O(t\theta))\end{aligned}$$

and analogously for p.

If $t = \mathsf{Q}\langle\tau\rangle\,(\lambda x.\, u)$, then

$$\begin{aligned}\mathcal{P}(O(t))\rho &= \mathsf{Q}_1(\mathcal{P}(O(\tau))\rho, \mathcal{P}(O(\mathcal{B}_x(u)))\rho)\\ &\stackrel{\text{IH}}{=} \mathsf{Q}_1(\mathcal{P}(O(\tau\theta)), \mathcal{P}(O(\mathcal{B}_x(u)\theta)))\\ &= \mathsf{Q}_1(\mathcal{P}(O(\tau\theta)), \mathcal{P}(O(\mathcal{B}_x(u\theta[x \mapsto x]))))\\ &= \mathcal{P}(O(\mathsf{Q}\langle\tau\rangle\,(\lambda x.\,(u\theta[x \mapsto x])))) = \mathcal{P}(O(\mathsf{Q}\langle\tau\rangle\,((\lambda x.\, u)\theta))) = \mathcal{P}(O(t\theta))\end{aligned}$$

and similarly for p, using Q'_1 instead of Q_1.

If $t = (\lambda x : \tau.\ u)$ and t is not fluid, then

$$\begin{aligned}\mathcal{P}(\mathcal{O}(t))\rho &= \mathsf{lam}(\mathcal{P}(\mathcal{O}(\tau))\rho, p(\mathcal{O}(\mathcal{B}_x(u)))\rho)\\ &\stackrel{\text{IH}}{=} \mathsf{lam}(\mathcal{P}(\mathcal{O}(\tau\theta)), p(\mathcal{O}(\mathcal{B}_x(u)\theta)))\\ &= \mathsf{lam}(\mathcal{P}(\mathcal{O}(\tau\theta)), p(\mathcal{O}(\mathcal{B}_x(u\theta[x \mapsto x]))))\\ &= \mathcal{P}(\mathcal{O}(\lambda x : \tau\theta.\ (u\theta[x \mapsto x]))) = \mathcal{P}(\mathcal{O}((\lambda x : \tau.\ u)\theta)) = \mathcal{P}(\mathcal{O}(t\theta))\end{aligned}$$

and analogously for p. □

4 Refutational Completeness

We present a proof of refutational completeness for our higher-order logic superposition calculus. The literature contains two different notions of refutational completeness: static and dynamic. They are defined as follows. For the precise definitions of inference systems and redundancy criteria, we refer to Waldmann et al. [35].

Definition 47 (Static refutational completeness). Let $\models$ be an entailment relation, let *Inf* be an inference system, and let $(\mathit{Red}_{\mathrm{I}}, \mathit{Red}_{\mathrm{C}})$ be a redundancy criterion. The inference system *Inf* is *statically refutationally complete* w.r.t. $\models$ and $(\mathit{Red}_{\mathrm{I}}, \mathit{Red}_{\mathrm{C}})$ if we have $N \models \bot$ if and only if $\bot \in N$ for every clause set N that is saturated w.r.t. *Inf* and $\mathit{Red}_{\mathrm{I}}$.

Definition 48 (Dynamic refutational completeness). Let $\models$ be an entailment relation, let *Inf* be an inference system, and let $(\mathit{Red}_{\mathrm{I}}, \mathit{Red}_{\mathrm{C}})$ be a redundancy criterion. Let $(N_i)_i$ be a finite or infinite sequence over sets of clauses. Such a sequence is a *derivation* if $N_i \setminus N_{i+1} \subseteq \mathit{Red}_{\mathrm{C}}(N_{i+1})$ for all i. It is *fair* if all *Inf*-inferences from clauses in the limit inferior $\bigcup_i \bigcap_{j \geq i} N_j$ are contained in $\bigcup_i \mathit{Red}_{\mathrm{I}}(N_i)$. The inference system *Inf* is *dynamically refutationally complete* w.r.t. $\models$ and $(\mathit{Red}_{\mathrm{I}}, \mathit{Red}_{\mathrm{C}})$ if for every fair derivation $(N_i)_i$ such that $N_0 \models \bot$, we have $\bot \in N_i$ for some i.

We have introduced three different notions of entailment on the H level: $\models_{\mathcal{G}}$, $\models$, and $\models_{\approx}$. With respect to $\models_{\mathcal{G}}$, static and dynamic completeness hold unconditionally. For the other two notions of entailment, we will need to add an additional precondition that ensure that the initial clause set is $\mathsf{Q}_{\approx}$-normal, which can only be stated for dynamic completeness. For $\models_{\approx}$, we need to require in addition that the initial clause set does not contain any sk symbols.

4.1 Outline of the Proof

Following the completeness proof of λfSup and λSup, our proof proceeds in three steps, corresponding to the three levels defined above:

1. We prove static refutational completeness of *GFInf*.
2. We show static refutational completeness of *GHInf* by transforming the model constructed on the GF level into a higher-order interpretation.

3. We lift the result to the H level by invoking the saturation framework of Waldmann et al. [35].

For the first step, since *GFInf* is essentially identical with oSup, we can rely on Nummelin et al.'s completeness theorem for oSup. The refutational completeness result holds for any tuple of parameters $q \in \mathcal{F}(Q)$. In addition to the refutational completeness of *GFInf*, the subsequent steps also depend on some properties of the constructed model, which we can easily derive from Lemmas proven by Nummelin et al.

For the second step, we fix a parameter triple $q \in Q$ and a set $N \subseteq \mathcal{C}_{\mathrm{GH}}$ saturated w.r.t. GHInf^q and not containing the empty clause. Then the first step guarantees us a model of $\mathcal{F}(N)$. Based on this model, we construct a higher-order interpretation that we show to be a model of N. In essence, the proof is analogous to the one of λSup, but additionally, we need to consider $\mathsf{Q}_{\approx}$-normality and the logical symbols.

For the third step, the main proof obligation the saturation framework leaves to us is to show that nonredundant GHInf^q inferences can be lifted to corresponding *HInf* inferences. For this lifting, we must choose a suitable parameter triple $q \in Q$, given a clause set $N \subseteq \mathcal{C}_{\mathrm{H}}$ saturated w.r.t. *HInf* and the H selection functions. In particular, we must specify the witness function to produce Skolem terms according to the given set N. Then the saturation framework guarantees static refutational completeness w.r.t. $\models_{\mathcal{G}}$. We show that this implies dynamic refutational completeness w.r.t. $\models$ for $\mathsf{Q}_{\approx}$-normal initial clause sets.

4.2 The Ground First-Order Level

The inference system *GFInf* is essentially identical with Nummelin et al.'s ground oSup calculus, with the following caveats:

- The conditions of EFACT superficially appear different but are equivalent.
- *GFInf*'s last condition of FORALLRW and EXISTSRW is weaker than oSup's to simplify our presentation. Where oSup tests for general tautologies, *GFInf* only tests for a certain form of tautological literal. Clearly, this does not compromise the refutational completeness of *GFInf*.
- The redundancy criterion of oSup requires that a finite subset of $\{D \in N \mid D \prec C\}$ entails C, whereas $\mathit{GFRed}_{\mathrm{C}}$ requires that $\{D \in N \mid D \prec C\}$ entails C. Since first-order logic with Booleans is compact, the two criteria are equivalent.

Similar to Bachmair and Ganzinger's construction for Sup, Nummelin et al. define the model via a term rewriting system R_M^*. As they explain in Definition 16 and Lemma 17, such a term rewriting system R_M^* can be viewed as a interpretation in GF's logic with the property that $t \leftrightarrow_{R_M^*} s$ if and only if $[\![t]\!]_{R_M^*} = [\![s]\!]_{R_M^*}$ for any two ground terms t and s.

The interpretation R_M^* fulfills the following property:

Lemma 49. *Let $N \subseteq \mathcal{C}_{\mathrm{GF}}$. If $C \in N$ where $C = C' \vee s \approx t$ produces $s \rightarrow t$, then $s \approx t$ is strictly $\succeq$-eligible in C and C' is false in R_M^*.*

Proof. The literal $s \approx t$ is $\succeq$-eligible in C by (C3) of oSup It is even strictly $\succeq$-eligible by (C6). The subclause C' is false in R_M^* by Lemma 26 of oSup.

Since Nummelin et al. prove refutational completeness, but do not explicitly state that $R^*_{N\setminus GFRed_{\mathrm{C}}(N)}$ is the model witnessing the completeness, we retrace the last step of their completeness proof in the proof of the following theorem. This also allows us to circumvent the mismatch between the redundancy criteria. Adapted to the context of this report, their completeness theorem can be restated as follows:

Theorem 50 (Ground first-order static refutational completeness). *Let $q \in \mathcal{F}(Q)$. Then the inference system $GFInf^q$ is statically refutationally complete w.r.t. $\models$ and $(GFRed_{\mathrm{I}}, GFRed_{\mathrm{C}})$. More precisely, if $N \subseteq \mathcal{C}_{\mathrm{GF}}$ is a clause set saturated w.r.t. $GFInf^q$ and $GFRed^q_{\mathrm{I}}$ such that $\bot \notin N$, then $R^*_{N\setminus GFRed_{\mathrm{C}}(N)}$ is a model of N.*

Proof. This proof is inspired by the one of Theorem 4.9 of Bachmair and Ganzinger [3].

By Lemma 31 of oSup, $GFInf^q$ fulfills the reduction property of counterexamples w.r.t. $\succ$. This means that for any clause set M where C is the smallest clause in M that is false in R^*_M, there exists an inference from M with

- main premise C,
- side premises that are true in R^*_M, and
- a conclusion that is smaller than C and false in R^*_M.

To derive a contradiction, we assume that $R^*_{N\setminus GFRed_{\mathrm{C}}(N)} \not\models N$. Then there must be a smallest clause in $C \in N$ that is false in $R^*_{N\setminus GFRed_{\mathrm{C}}(N)}$. Using $M = N \setminus GFRed_{\mathrm{C}}(N)$, we obtain an inference ι with the above properties. Since N is saturated w.r.t. $GFInf^q$ and $GFRed^q_{\mathrm{I}}$ and $prems(\iota) \subseteq M \subseteq N$, we have $\iota \in GFRed^q_{\mathrm{I}}(N)$. By definition, this means $prems(\iota) \cap GFRed_{\mathrm{C}}(N) \neq \varnothing$ or $\{D \in N \mid D \prec C\} \models concl(\iota)$. Since $prems(\iota) \subseteq M = N \setminus GFRed_{\mathrm{C}}(N)$, it must be the latter. Then $R^*_M \not\models \{D \in N \mid D \prec C\}$ because $R^*_M \not\models concl(\iota)$. This contradicts the minimality of C. □

4.3 The Ground Higher-Order Level

In this subsection, let $q = (GHLitSel, GHBoolSel, GHWit) \in Q$ be a parameter triple and let $N \subseteq \mathcal{C}_{\mathrm{GH}}$. Since all terms on the GH level are $\mathsf{Q}_{\approx}$-normal, in particular N is $\mathsf{Q}_{\approx}$-normal. We assume that N is saturated w.r.t. $GHInf^q$ and $GHRed^q_{\mathrm{I}}$, and that $\bot \notin N$. Clearly, $\mathcal{F}(N)$ is then saturated w.r.t. $GFInf^{\mathcal{F}(sel)}$ and $GFRed^{\mathcal{F}(sel)}_{\mathrm{I}}$ and $R^*_{\mathcal{F}(N)\setminus GFRed_{\mathrm{C}}(N)}$ is a model of $\mathcal{F}(N)$ by Theorem 50.

In the following, we abbreviate $R^*_{\mathcal{F}(N)\setminus GFRed_{\mathrm{C}}(N)}$ as R. Given two terms $s, t \in \mathcal{T}_{\mathrm{GH}}$, we write $s \sim t$ to abbreviate $R \models \mathcal{F}(s) \approx \mathcal{F}(t)$, which is equivalent to $[\![\mathcal{F}(s)]\!]_R = [\![\mathcal{F}(t)]\!]_R$.

Lemma 51. *For all terms $t, s : \tau \to \upsilon$ in $\mathcal{T}_{\mathrm{GH}}$, these statements are equivalent:*

1. $t \sim s$;
2. $t\,(\mathsf{diff}\,t\,s) \sim s\,(\mathsf{diff}\,t\,s)$;
3. $t\,u \sim s\,u$ *for all* $u \in \mathcal{T}_{\mathrm{GH}}$.

Proof. Analogous to Lemma 38 of λSup, using Lemma 49 and the $\beta\eta\mathsf{Q}_{\eta}$-normal form. □

Lemma 52. *Let $s \in \mathcal{T}_{\mathrm{H}}$ and let θ, θ' be grounding substitutions such that $s\theta$ and $s\theta'$ are $\mathsf{Q}_{\approx}$-normal, $x\theta \sim x\theta'$ for all variables x, and $\alpha\theta = \alpha\theta'$ for all type variables α. Then $s\theta \sim s\theta'$.*

Proof. This proof is almost identical to the one of Lemma 39 of λSup [7]. The only difference lies in Case 4.1 that must deal with quantifiers. What follows is a copy of the beginning of this previous proof that introduces the notions relevant to deal with the extra case here.

In this proof, we work directly on λ-terms. To prove the lemma, it suffices to prove it for any λ-term s. Here, for λ-terms t_1 and t_2, the notation $t_1 \sim t_2$ is to be read as $t_1{\downarrow_{\beta\eta\mathsf{Q}_\eta}} \sim t_2{\downarrow_{\beta\eta\mathsf{Q}_\eta}}$ because $\mathcal{F}$ is only defined on $\beta\eta\mathsf{Q}_\eta$-normal terms.

DEFINITION We extend the syntax of λ-terms with a new polymorphic function symbol $\oplus : \Pi\alpha.\ \alpha \to \alpha \to \alpha$. We will omit its type argument. It is equipped with two reduction rules: $\oplus\, t\, s \to t$ and $\oplus\, t\, s \to s$. A $\beta\oplus$*-reduction step* is either a rewrite step following one of these rules or a β-reduction step.

The computability path order $\succ_{\mathsf{CPO}}$ [11] guarantees that

- $\oplus\, t\, s \succ_{\mathsf{CPO}} s$ by applying rule $@\rhd$;
- $\oplus\, t\, s \succ_{\mathsf{CPO}} t$ by applying rule $@\rhd$ twice;
- $(\lambda x.\, t)\, s \succ_{\mathsf{CPO}} t[x \mapsto s]$ by applying rule $@\beta$.

Since this order is moreover monotone, it decreases with $\beta\oplus$-reduction steps. The order is also well-founded; thus, $\beta\oplus$-reductions terminate. And since the $\beta\oplus$-reduction steps describe a finitely branching term rewrite system, by Kőnig's lemma [23], there is a maximal number of $\beta\oplus$-reduction steps from each λ-term.

DEFINITION A λ-term is *term-ground* if it does not contain free term variables. It may contain polymorphic type arguments.

DEFINITION We introduce an auxiliary function $\mathcal{S}$ that essentially measures the size of a λ-term but assigns a size of 1 to term-ground λ-terms.

$$\mathcal{S}(s) = \begin{cases} 1 & \text{if } s \text{ is term-ground or is a bound or free variable or a symbol} \\ 1 + \mathcal{S}(t) & \text{if } s \text{ is not term-ground and has the form } \lambda x.\, t \\ \mathcal{S}(t) + \mathcal{S}(u) & \text{if } s \text{ is not term-ground and has the form } t\, u \end{cases}$$

We prove $s\theta \sim s\theta'$ by well-founded induction on s, θ, and θ' using the left-to-right lexicographic order on the triple $\big(n_1(s), n_2(s), n_3(s)\big) \in \mathbb{N}^3$, where

- $n_1(s)$ is the maximal number of $\beta\oplus$-reduction steps starting from $s\sigma$, where σ is the substitution mapping each term variable x to $\oplus\, x\theta\, x\theta'$;
- $n_2(s)$ is the number of free term variables occurring more than once in s;
- $n_3(s) = \mathcal{S}(s)$.

In Case 4.1 of the proof of Lemma 39 of λSup [7], we consider a λ-term s that contains exactly one free term-variable that occurs exactly once in s and s is of the form $f\langle\bar{\tau}\rangle\, \bar{t}$, for some symbol f, some types $\bar{\tau}$, and some λ-terms $\bar{t}$. For any $f \notin \{\forall, \exists\}$

the proof proceeds as before, because then the definition of the encoding $\mathcal{F}$ coincides with the one of λSup. The remaining case to handle is thus when s is of the form $\mathsf{Q}\langle\tau\rangle\,(\lambda z.\,t)$.

In that case, we have $[\![\mathcal{F}(s\theta)]\!]_R = [\![\mathrm{Q}z.\,\mathcal{F}(t\theta)]\!]_R$ where $\mathrm{Q} \in \{\forall,\exists\}$, but θ and θ' are not necessarily grounding for t since t may contain the variable z that is bound in s. Thus we cannot apply our induction hypothesis directly on t. Instead we want to apply it on $t\{z \mapsto u\}$, which we denote t_u, where $u \in \mathcal{T}_{\mathrm{GH}}$. It is possible to apply the induction hypothesis to obtain $t_u\theta \sim t_u\theta'$ because

- $n_1(s) = n_1(t_u)$ since all $\beta\oplus$-reductions in s are also in t_u;
- $n_2(s) = 0 = n_2(t_u)$ since the same unique free term variable occurs in s and in t_u; and
- $n_3(t_u) < n_3(s)$ because $\mathcal{S}(z) = \mathcal{S}(u) = 1$ implies $n_3(t_u) = n_3(t)$ and hence $n_3(s) = \mathcal{S}(\mathsf{Q}) + \mathcal{S}(\lambda z.\,t) = 1+1+\mathcal{S}(t) = 1+1+\mathcal{S}(t_u)$.

Moreover, since θ and θ' do not capture z, and since u is $\mathsf{Q}_{\approx}$-normal, $t_u\theta = (t\theta)\{z \mapsto u\}$ is $\mathsf{Q}_{\approx}$-normal by Lemma 9 and similarly for θ'. Thus we obtain $(t\theta)\{z \mapsto u\} \sim (t\theta')\{z \mapsto u\}$.

Now, let us consider the case where $\mathsf{Q} = \boldsymbol{\forall}$: By the definition of interpretations on the GF level, by Lemma 6 (substitution lemma) of oSup, and R being term-generated, we have

$$\begin{aligned}
[\![\forall z.\,\mathcal{F}(t\theta)]\!]_R^{\xi} &= \min\{[\![\mathcal{F}(t\theta)]\!]_R^{\xi[z\mapsto a]} \mid a \in \mathcal{U}_\tau\} \\
&= \min\{[\![\mathcal{F}(t\theta)\{z \mapsto v\}]\!]_R^{\xi} \mid v \in \mathcal{T}_{\mathrm{GF}}\} \\
&= \min\{[\![\mathcal{F}(t\theta\{z \mapsto u\})]\!]_R^{\xi} \mid u \in \mathcal{T}_{\mathrm{GH}}\}
\end{aligned}$$

The same holds for θ'. Moreover, above we deduced $(t\theta)\{z \mapsto u\} \sim (t\theta')\{z \mapsto u\}$ from the induction hypothesis. Thus, $s\theta \sim s\theta'$, as desired in the case $\mathsf{Q} = \boldsymbol{\forall}$. The case $\mathsf{Q} = \boldsymbol{\exists}$ is analogous, with max instead of min. □

We proceed by defining a higher-order interpretation $\mathcal{I}^{\mathrm{GH}} = (\mathcal{U}^{\mathrm{GH}}, \mathcal{J}_{\mathsf{ty}}^{\mathrm{GH}}, \mathcal{J}^{\mathrm{GH}}, \mathcal{L}^{\mathrm{GH}})$ derived from R. We call this interpretation $\mathcal{I}^{\mathrm{GH}}$ because we use it to show refutational completeness of *GHInf*; it is a higher-order interpretation as defined in Section 2, which can interpret ground as well as nonground terms. Let $(\mathcal{U}, \mathcal{J}) = R$, meaning that $\mathcal{U}_\tau$ is the universe of R for type τ and $\mathcal{J}$ is the interpretation function of R.

To illustrate the construction, we will employ the following running example. Let $\Sigma_{\mathsf{ty}} = \{\iota, o, \to\}$ and let Σ contain $\mathsf{f} : \iota \to \iota$ and $\mathsf{a} : \iota$, as well as the logical symbols and the choice constant ε. Then, on the GF level, the type signature is also Σ_{ty}, and the term signature is the set Σ_{GF}, which contains f_0, f_1, a_0, subscripted versions of all logical symbols, such as $\top_0$, $\bot_0$, $\neg_0$, and $\neg_1$, as well as symbols ε_i^τ for each $\tau \in \mathcal{Ty}_{\mathrm{GH}}$, and a symbol $\mathsf{lam}_{\lambda x.\,t}$ for each $\lambda x.\,t \in \mathcal{T}_{\mathrm{GH}}$. We write $[t]$ for the equivalence class of $t \in \mathcal{T}_{\mathrm{GF}}$ modulo R. The universes $\mathcal{U}_\tau$ are sets of such equivalence classes; for instance $[\mathsf{f}_1(\mathsf{a}_0)] \in \mathcal{U}_\iota$, $[\neg_0] \in \mathcal{U}_{o\to o}$, and $[\mathsf{lam}_{\lambda x.\,\mathsf{a}}] \in \mathcal{U}_{\iota\to\iota}$. We assume that R is such that $[\mathsf{a}_0]$, $[\mathsf{f}_1(\mathsf{a}_0)]$, $[\mathsf{f}_1(\mathsf{f}_1(\mathsf{a}_0))], \ldots$ are all different from each other, and therefore that $\mathcal{U}_\iota$ is infinite.

When defining the universe $\mathcal{U}^{\mathrm{GH}}$ of the higher-order interpretation, we need to ensure that it contains subsets of function spaces, since $\mathcal{J}_{\mathsf{ty}}^{\mathrm{GH}}(\to)(\mathcal{D}_1, \mathcal{D}_2)$ must be a subset of the function space from $\mathcal{D}_1$ to $\mathcal{D}_2$ for all $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{U}^{\mathrm{GH}}$. However, the first-order universes $\mathcal{U}_\tau$ consist of equivalence classes of terms from $\mathcal{T}_{\mathrm{GF}}$ w.r.t. the rewriting system R, not of functions.

To repair this mismatch, we will define a family of functions $\mathcal{E}_\tau$ that give a meaning to the elements of the first-order universes $\mathcal{U}_\tau$. We will define a domain $\mathcal{D}_\tau$ for each ground type τ and then let $\mathcal{U}^{\mathrm{GH}}$ be the set of all these domains $\mathcal{D}_\tau$. Thus, there will be a one-to-one correspondence between ground types and domains. Since the higher-order and first-order type signatures are identical (including $\to$, which is uninterpreted in GF's logic), we can identify higher-order and first-order types.

We define $\mathcal{E}_\tau$ and $\mathcal{D}_\tau$ in a mutual recursion. To ensure well definedness, we must simultaneously show that $\mathcal{E}_\tau$ is bijective. We start with nonfunctional types τ: Let $\mathcal{D}_\tau = \mathcal{U}_\tau$ and let $\mathcal{E}_\tau : \mathcal{U}_\tau \longrightarrow \mathcal{D}_\tau$ be the identity. Clearly, the identity is bijective. For functional types, we define

$$\mathcal{D}_{\tau\to\upsilon} = \big\{\varphi : \mathcal{D}_\tau \longrightarrow \mathcal{D}_\upsilon \mid \exists\, s : \tau \to \upsilon.\ \forall\, u : \tau.\ \varphi\big(\mathcal{E}_\tau\big([\![\mathcal{F}(u)]\!]_R\big)\big) = \mathcal{E}_\upsilon\big([\![\mathcal{F}(s\,u)]\!]_R\big)\big\}$$

$$\mathcal{E}_{\tau\to\upsilon} : \mathcal{U}_{\tau\to\upsilon} \longrightarrow \mathcal{D}_{\tau\to\upsilon}$$

$$\mathcal{E}_{\tau\to\upsilon}([\![\mathcal{F}(s)]\!]_R)\big(\mathcal{E}_\tau\big([\![\mathcal{F}(u)]\!]_R\big)\big) = \mathcal{E}_\upsilon\big([\![\mathcal{F}(s\,u)]\!]_R\big)$$

To verify that this equation is a valid definition of $\mathcal{E}_{\tau\to\upsilon}$, we must show that

- every element of $\mathcal{U}_{\tau\to\upsilon}$ is of the form $[\![\mathcal{F}(s)]\!]_R$ for some $s \in \mathcal{T}_{\mathrm{GH}}$;
- every element of $\mathcal{D}_\tau$ is of the form $\mathcal{E}_\tau\big([\![\mathcal{F}(u)]\!]_R\big)$ for some $u \in \mathcal{T}_{\mathrm{GH}}$;
- the definition does not depend on the choice of such s and u;
- $\mathcal{E}_{\tau\to\upsilon}([\![\mathcal{F}(s)]\!]_R) \in \mathcal{D}_{\tau\to\upsilon}$ for all $s \in \mathcal{T}_{\mathrm{GH}}$.

The first claim holds because R is term-generated and $\mathcal{F}$ is a bijection. The second claim holds because R is term-generated and $\mathcal{F}$ and $\mathcal{E}_\tau$ are bijections. To prove the third claim, we assume that there are other ground terms $t \in \mathcal{T}_{\mathrm{GH}}$ and $v \in \mathcal{T}_{\mathrm{GH}}$ such that $[\![\mathcal{F}(s)]\!]_R = [\![\mathcal{F}(t)]\!]_R$ and $\mathcal{E}_\tau\big([\![\mathcal{F}(u)]\!]_R\big) = \mathcal{E}_\tau\big([\![\mathcal{F}(v)]\!]_R\big)$. Since $\mathcal{E}_\tau$ is bijective, we have $[\![\mathcal{F}(u)]\!]_R = [\![\mathcal{F}(v)]\!]_R$. Using the $\sim$-notation, we can write this as $u \sim v$. The terms s, t, u, and v are in $\mathcal{T}_{\mathrm{GH}}$, and thus $\mathsf{Q}_\approx$-normal, allowing us to apply Lemma 52 to the term $x\,y$ and the substitutions $\{x \mapsto s, y \mapsto u\}$ and $\{x \mapsto t, y \mapsto v\}$. Thus, we obtain $s\,u \sim t\,v$—i.e., $[\![\mathcal{F}(s\,u)]\!]_R = [\![\mathcal{F}(t\,v)]\!]_R$, indicating that the definition of $\mathcal{E}_{\tau\to\upsilon}$ above does not depend on the choice of s and u. The fourth claim is obvious from the definition of $\mathcal{D}_{\tau\to\upsilon}$ and the third claim.

It remains to show that $\mathcal{E}_{\tau\to\upsilon}$ is bijective. For injectivity, we fix two terms $s, t \in \mathcal{T}_{\mathrm{GH}}$ such that for all $u \in \mathcal{T}_{\mathrm{GH}}$, we have $[\![\mathcal{F}(s\,u)]\!]_R = [\![\mathcal{F}(t\,u)]\!]_R$. By Lemma 51, it follows that $[\![\mathcal{F}(s)]\!]_R = [\![\mathcal{F}(t)]\!]_R$, which shows that $\mathcal{E}_{\tau\to\upsilon}$ is injective. For surjectivity, we fix an element $\varphi \in \mathcal{D}_{\tau\to\upsilon}$. By definition of $\mathcal{D}_{\tau\to\upsilon}$, there exists a term s such that $\varphi\big(\mathcal{E}_\tau\big([\![\mathcal{F}(u)]\!]_R\big)\big) = \mathcal{E}_\upsilon\big([\![\mathcal{F}(s\,u)]\!]_R\big)$ for all u. Hence, $\mathcal{E}_{\tau\to\upsilon}([\![\mathcal{F}(s)]\!]_R) = \varphi$, proving surjectivity and therefore bijectivity of $\mathcal{E}_{\tau\to\upsilon}$. Below, we will usually write $\mathcal{E}$ instead of $\mathcal{E}_\tau$ since the type τ is determined by $\mathcal{E}_\tau$'s first argument.

In our running example, we thus have $\mathcal{D}_\iota = \mathcal{U}_\iota = \{[\mathsf{a}_0], [\mathsf{f}_1(\mathsf{a}_0)], [\mathsf{f}_1(\mathsf{f}_1(\mathsf{a}_0))], \ldots\}$ and $\mathcal{E}_\iota$ is the identity $\mathcal{U}_\iota \to \mathcal{D}_\iota,\ c \mapsto c$. The function $\mathcal{E}^0_{\iota\to\iota}$ maps $[\mathsf{lam}_{\lambda x.x}]$ to the identity $\mathcal{D}_\iota \to \mathcal{D}_\iota,\ c \mapsto c$; it maps $[\mathsf{lam}_{\lambda x.\mathsf{a}}]$ to the constant function $\mathcal{D}_\iota \to \mathcal{D}_\iota,\ c \mapsto [\mathsf{a}_0]$; it maps $[\mathsf{f}_0]$ to the function $\mathcal{D}_\iota \to \mathcal{D}_\iota,\ [t] \mapsto [\mathsf{f}_1(t)]$; and it maps $[\varepsilon_1^{\iota\to\iota}(\mathsf{lam}_{\lambda z.\, z\,(\mathsf{f\,a}) \approx \mathsf{a}})]$ to the function

$\mathcal{D}_\iota \to \mathcal{D}_\iota$, $[t] \mapsto [\varepsilon_2^{\iota\to\iota}(\mathsf{lam}_{\lambda z.\, z\,(\mathsf{f}\,\mathsf{a})\,\boldsymbol{\approx}\,\mathsf{a}}, t)]$. There are a lot more different functions in $\mathcal{D}_{\iota\to\iota} = \mathcal{E}^0_{\iota\to\iota}(\mathcal{U}_{\iota\to\iota})$, but still $\mathcal{D}_{\iota\to\iota}$ is a proper subset of the function space $\mathcal{U}_\iota \to \mathcal{U}_\iota$ because the function space is uncountably infinite, whereas $\mathcal{T}_{\mathrm{GF}}$ and hence $\mathcal{D}_{\iota\to\iota}$ is countable. Thus, the construction works only because we allow nonstandard Henkin models.

We define the higher-order universe as $\mathcal{U}^{\mathrm{GH}} = \{\mathcal{D}_\tau \mid \tau \text{ ground}\}$. In particular, this implies that $\mathcal{D}_{\mathrm{o}} = \{0,1\} \in \mathcal{U}^{\mathrm{GH}}$ as needed, where 0 is identified with $[\bot_0]$ and 1 with $[\top_0]$. Moreover, we define $\mathcal{J}^{\mathrm{GH}}_{\mathrm{ty}}(\kappa)(\mathcal{D}_{\bar{\tau}}) = \mathcal{U}_{\kappa(\bar{\tau})}$ for all $\kappa \in \Sigma_{\mathrm{ty}}$, completing the type interpretation of $\mathcal{I}^{\mathrm{GH}}_{\mathrm{ty}} = (\mathcal{U}^{\mathrm{GH}}, \mathcal{J}^{\mathrm{GH}}_{\mathrm{ty}})$ and ensuring that $\mathcal{J}^{\mathrm{GH}}_{\mathrm{ty}}(\mathrm{o}) = \mathcal{U}_{\mathrm{o}} = \{0,1\}$.

We define the interpretation function $\mathcal{J}^{\mathrm{GH}}$ for nonquantifier symbols $\mathsf{f} : \Pi\bar{\alpha}_m.\, \tau$ by $\mathcal{J}^{\mathrm{GH}}(\mathsf{f}, \mathcal{D}_{\bar{\upsilon}_m}) = \mathcal{E}(\mathcal{J}(\mathsf{f}_0^{\bar{\upsilon}_m}))$, and for quantifiers by $\mathcal{J}^{\mathrm{GH}}(\boldsymbol{\forall}, \mathcal{D}_\tau)(f) = \min\{f(a) \mid a \in \mathcal{D}_\tau\}$ and $\mathcal{J}^{\mathrm{GH}}(\boldsymbol{\exists}, \mathcal{D}_\tau)(f) = \max\{f(a) \mid a \in \mathcal{D}_\tau\}$ for all $f \in \mathcal{J}^{\mathrm{GH}}_{\mathrm{ty}}(\to)(\mathcal{D}_\tau, \{0,1\})$.

In our example, we thus have $\mathcal{J}^{\mathrm{GH}}(\mathsf{f}) = \mathcal{E}([\mathsf{f}_0])$, which is the function $[t] \mapsto [\mathsf{f}_1(t)]$.

We must show that this definition indeed fulfills the requirements of an interpretation function. By definition, we have (I1) $\mathcal{J}^{\mathrm{GH}}(\top) = \mathcal{E}([\![\top_0]\!]_R) = [\![\top_0]\!]_R = 1$ and (I2) $\mathcal{J}^{\mathrm{GH}}(\bot) = \mathcal{E}([\![\bot_0]\!]_R) = [\![\bot_0]\!]_R = 0$.

Let $a, b \in \{0,1\}$, $u_0 = \bot$, and $u_1 = \top$. Then

$$\begin{aligned}
&\text{(I3)}\ \ \mathcal{J}^{\mathrm{GH}}(\boldsymbol{\wedge})(a,b) = \mathcal{E}^0([\![\mathcal{F}(\boldsymbol{\wedge})]\!]_R)([\![\mathcal{F}(u_a)]\!]_R, [\![\mathcal{F}(u_b)]\!]_R)\\
&\qquad\qquad\qquad\ \ = \mathcal{E}([\![\mathcal{F}(u_a \boldsymbol{\wedge} u_b)]\!]_R) = \min\{a,b\}\\
&\text{(I4)}\ \ \mathcal{J}^{\mathrm{GH}}(\boldsymbol{\vee})(a,b) = \mathcal{E}([\![\mathcal{F}(u_a \boldsymbol{\vee} u_b)]\!]_R) = \max\{a,b\}\\
&\text{(I5)}\ \ \ \ \mathcal{J}^{\mathrm{GH}}(\boldsymbol{\neg})(a) = \mathcal{E}(\mathcal{J}(\mathcal{F}(\boldsymbol{\neg})))([\![\mathcal{F}(u_a)]\!]_R) = \mathcal{E}^0([\![\neg_0]\!]_R)([\![\mathcal{F}(u_a)]\!]_R)\\
&\qquad\qquad\qquad\ \ = \mathcal{E}([\![\mathcal{F}(\boldsymbol{\neg} u_a)]\!]_R) = [\![\mathcal{F}(\boldsymbol{\neg} u_a)]\!]_R = 1 - a\\
&\text{(I6)}\ \mathcal{J}^{\mathrm{GH}}(\boldsymbol{\rightarrow})(a,b) = \mathcal{E}([\![\mathcal{F}(u_a \boldsymbol{\rightarrow} u_b)]\!]_R) = \max\{1-a, b\}
\end{aligned}$$

Let $\mathcal{D} \in \mathcal{U}^{\mathrm{GH}}$ and $a', b' \in \mathcal{D}$. (I7) Since $\mathcal{E}$ is bijective and R is term-generated, there exist ground terms u and v such that $\mathcal{E}([\![\mathcal{F}(u)]\!]_R) = a'$ and $\mathcal{E}([\![\mathcal{F}(v)]\!]_R) = b'$. Then

$$\mathcal{J}^{\mathrm{GH}}(\boldsymbol{\approx})(a',b') = \mathcal{E}^0([\![\mathcal{F}(\boldsymbol{\approx})]\!]_R)(\mathcal{E}([\![\mathcal{F}(u)]\!]_R), \mathcal{E}([\![\mathcal{F}(v)]\!]_R)) = \mathcal{E}([\![\mathcal{F}(u \boldsymbol{\approx} v)]\!]_R)$$

which is 1 if $a' = b'$ and 0 otherwise. (I8) Similarly $\mathcal{J}^{\mathrm{GH}}(\boldsymbol{\not\approx})(a',b') = 0$ if $a' = b'$ and 1 otherwise.

(I9) (I10) The requirements for $\boldsymbol{\forall}$ and $\boldsymbol{\exists}$ hold by definition of $\mathcal{J}^{\mathrm{GH}}$.

(I11) Let $\mathcal{D}_\tau \in \mathcal{U}^{\mathrm{GH}}$ and $f \in \mathcal{J}_{\mathrm{ty}}(\to)(\mathcal{D}_\tau, \{0,1\})$. For the requirement on ε, we must show that $f(\mathcal{J}^{\mathrm{GH}}(\varepsilon\langle\tau\rangle)(f)) = \max\{f(a) \mid a \in \mathcal{D}_\tau\}$.

First, we assume that $f(\mathcal{J}^{\mathrm{GH}}(\varepsilon\langle\tau\rangle)(f)) = 0$. We want to show that $\max\{f(a) \mid a \in \mathcal{D}_\tau\} = 0$. Let $a \in \mathcal{D}_\tau$. We have $f = \mathcal{E}([\![\mathcal{F}(p)]\!]_R)$ for some $p : \tau \to \mathrm{o}$ and $a = \mathcal{E}([\![\mathcal{F}(u)]\!]_R)$ for some $u : \tau \in \mathcal{T}_{\mathrm{GH}}$ because $\mathcal{E}$ and $\mathcal{F}$ are bijective and R is term-generated. Since N is saturated, all conclusions of GCHOICE belong to N. In particular, we have $(p\,u \approx \bot \vee p\,(\varepsilon\langle\tau\rangle\,p) \approx \top) \in N$ and hence $[\![\mathcal{F}(p\,u \approx \bot \vee p\,(\varepsilon\langle\tau\rangle\,p) \approx \top)]\!]_R = 1$. Thus, we have $\max\{[\![\mathcal{F}(p\,u \approx \bot)]\!]_R,\ [\![\mathcal{F}(p\,(\varepsilon\langle\tau\rangle\,p) \approx \top)]\!]_R\} = 1$. Since we have $f(\mathcal{J}^{\mathrm{GH}}(\varepsilon\langle\tau\rangle)(f)) =$

$[\![\mathcal{F}(p\,(\varepsilon\langle\tau\rangle\,p) \approx \top)]\!]_R$, our assumption implies that $[\![\mathcal{F}(p\,u \approx \bot)]\!]_R = 1$. Moreover,

$$\begin{aligned}
[\![\mathcal{F}(p\,u \approx \bot)]\!]_R = 1 \quad &\Leftrightarrow \quad [\![(\mathcal{F}(p\,u)) \approx \bot_0]\!]_R = 1 \\
&\Leftrightarrow \quad [\![\mathcal{F}(p\,u)]\!]_R = 0 \\
&\Leftrightarrow \quad \mathcal{E}([\![\mathcal{F}(p\,u)]\!]_R) = 0 \\
&\Leftrightarrow \quad \mathcal{E}([\![\mathcal{F}(p)]\!]_R)(\mathcal{E}([\![\mathcal{F}(u)]\!]_R)) = 0 \\
&\Leftrightarrow \quad f(a) = 0
\end{aligned}$$

Since our choice of a was arbitrary, this shows that $\max\{f(a) \mid a \in \mathcal{D}_\tau\} = 0$.

For the other direction, we assume that $\max\{f(a) \mid a \in \mathcal{D}_\tau\} = 0$. Then, by definition of max, and since $\mathcal{J}^{\mathrm{GH}}(\varepsilon\langle\tau\rangle)(f) \in \mathcal{D}_\tau$, we have in particular $f(\mathcal{J}^{\mathrm{GH}}(\varepsilon\langle\tau\rangle)(f)) = 0$. Thus, we have $f(\mathcal{J}^{\mathrm{GH}}(\varepsilon\langle\tau\rangle)(f)) = \max\{f(a) \mid a \in \mathcal{D}_\tau\}$ as required, which concludes the proof that $\mathcal{J}^{\mathrm{GH}}$ is an interpretation function.

Finally, we need to define the designation function $\mathcal{L}^{\mathrm{GH}}$, which takes a valuation ξ and a λ-expression as arguments. Given a valuation ξ, we choose a grounding substitution θ such that $\mathcal{D}_{\alpha\theta} = \xi(\alpha)$ and $\mathcal{E}([\![\mathcal{F}(x\theta)]\!]_R) = \xi(x)$ for all type variables α and all variables x. Such a substitution can be constructed as follows: We can fulfill the first equation in a unique way because there is a one-to-one correspondence between ground types and domains. Since $\mathcal{E}^{-1}(\xi(x))$ is an element of a first-order universe and R is term-generated, there exists a ground term s such that $[\![s]\!]_R^\xi = \mathcal{E}^{-1}(\xi(x))$. Choosing one such t and defining $x\theta = \mathcal{F}^{-1}(s)$ gives us a grounding substitution θ with the desired property.

We define $\mathcal{L}^{\mathrm{GH}}(\xi, (\lambda x.\, t)) = \mathcal{E}([\![\mathcal{F}((\lambda x.\, t)\theta)]\!]_R)$ if $\lambda x.\, t$ is $\mathsf{Q}_\approx$-normal, and otherwise $\mathcal{L}^{\mathrm{GH}}(\xi, (\lambda x.\, t)) = \mathcal{L}^{\mathrm{GH}}(\xi, (\lambda x.\, t)\downarrow_{\mathsf{Q}_\approx})$. Since $\mathcal{F}$ is only defined on $\mathsf{Q}_\approx$-normal terms, we need to show that $(\lambda x.t)\theta$ is $\mathsf{Q}_\approx$-normal if $\lambda x.\, t$ is $\mathsf{Q}_\approx$-normal. This holds because by construction of θ all $x\theta$ are $\mathsf{Q}_\approx$-normal and thus so is $(\lambda x.t)\theta$ according to Lemma 9. Moreover we need to show that our definition does not depend on the choice of θ. We assume that there exists another substitution θ' with the properties $\mathcal{D}_{\alpha\theta'} = \xi(\alpha)$ for all α and $\mathcal{E}([\![\mathcal{F}(x\theta')]\!]_R) = \xi(x)$ for all x. Then we have $\alpha\theta = \alpha\theta'$ for all α due to the one-to-one correspondence between domains and ground types. We have $[\![\mathcal{F}(x\theta)]\!]_R = [\![\mathcal{F}(x\theta')]\!]_R$ for all x because $\mathcal{E}$ is injective. By Lemma 52 it follows that $[\![\mathcal{F}((\lambda x.\, t)\theta)]\!]_R = [\![\mathcal{F}((\lambda x.\, t)\theta')]\!]_R$, which proves that $\mathcal{L}^{\mathrm{GH}}$ is well defined.

In our running example, for all ξ we have $\mathcal{L}^{\mathrm{GH}}(\xi, \lambda x.\, x) = \mathcal{E}([\mathsf{lam}_{\lambda x.\, x}])$, which is the identity. If $\xi(y) = [\mathsf{a}_0]$, then $\mathcal{L}^{\mathrm{GH}}(\xi, \lambda x.\, y) = \mathcal{E}([\mathsf{lam}_{\lambda x.\, \mathsf{a}}])$, which is the constant function $c \mapsto [\mathsf{a}_0]$.

This concludes the definition of the interpretation $\mathcal{I}^{\mathrm{GH}} = (\mathcal{U}^{\mathrm{GH}}, \mathcal{J}_{\mathsf{ty}}^{\mathrm{GH}}, \mathcal{J}^{\mathrm{GH}}, \mathcal{L}^{\mathrm{GH}})$. It remains to show that $\mathcal{I}^{\mathrm{GH}}$ is proper. We need a lemma:

Lemma 53. *Let $\mathcal{I}$ be an interpretation such that $[\![\lambda x.\, t]\!]_{\mathcal{I}}^\xi(a) = [\![t\downarrow_{\mathsf{Q}_\approx}]\!]_{\mathcal{I}}^{\xi[x\mapsto a]}$ for all λ-terms t and all valuations ξ. Then $\mathsf{Q}_\approx$-normalization preserves denotations of terms and truth of clauses w.r.t. $\mathcal{I}$.*

Proof. We must show $[\![t]\!]_{\mathcal{I}}^\xi = [\![t\downarrow_{\mathsf{Q}_\approx}]\!]_{\mathcal{I}}^\xi$ for all λ-terms t and all valuations ξ. We cannot work with $\beta\eta$-equivalence classes here because that would require the interpretation to be proper and thus result in a circular argument. We proceed by induction on the structure of t.

If $t = \lambda x.\, u$, by applying our assumption on $\mathcal{I}$ twice and because $\mathsf{Q}_\approx$-normalization is idempotent, we have for all a

$$\begin{aligned}\llbracket t\rrbracket_{\mathcal{I}}^{\xi}(a) &= \llbracket \lambda x.\, u\rrbracket_{\mathcal{I}}^{\xi}(a) = \llbracket u\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi[x\mapsto a]} = \llbracket u\downarrow_{\mathsf{Q}_\approx}\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi[x\mapsto a]} = \llbracket \lambda x.\,(u\downarrow_{\mathsf{Q}_\approx})\rrbracket_{\mathcal{I}}^{\xi}(a)\\ &= \llbracket (\lambda x.\, u)\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi}(a) = \llbracket t\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi}(a)\end{aligned}$$

Otherwise, if $t = s\,\bar{u}$, where s is a head that is not of the form $\mathsf{Q}\langle\tau\rangle$,

$$\llbracket t\rrbracket_{\mathcal{I}}^{\xi} = \llbracket s\,\bar{u}\rrbracket_{\mathcal{I}}^{\xi} = \llbracket s\rrbracket_{\mathcal{I}}^{\xi}(\llbracket \bar{u}\rrbracket_{\mathcal{I}}^{\xi}) \overset{\text{IH}}{=} \llbracket s\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi}(\llbracket \bar{u}\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi}) = \llbracket (s\,\bar{u})\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi} = \llbracket t\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi}$$

If $t = \mathsf{Q}\langle\tau\rangle\, u$, where t is not $\mathsf{Q}_\approx$-reducible at its head, then

$$\begin{aligned}\llbracket t\rrbracket_{\mathcal{I}}^{\xi} &= \llbracket \mathsf{Q}\langle\tau\rangle\, u\rrbracket_{\mathcal{I}}^{\xi} = \llbracket \mathsf{Q}\langle\tau\rangle\rrbracket_{\mathcal{I}}^{\xi}(\llbracket u\rrbracket_{\mathcal{I}}^{\xi}) \overset{\text{IH}}{=} \llbracket \mathsf{Q}\langle\tau\rangle\rrbracket_{\mathcal{I}}^{\xi}(\llbracket u\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi})\\ &= \llbracket (\mathsf{Q}\langle\tau\rangle\, u)\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi} = \llbracket t\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi}\end{aligned}$$

Otherwise we have $t = \mathsf{Q}\langle\tau\rangle$ or $t = \mathsf{Q}\langle\tau\rangle\, u$ such that t is $\mathsf{Q}_\approx$-reducible at its head. In these cases, it suffices to show that $\llbracket \forall\langle\tau\rangle\rrbracket_{\mathcal{I}}^{\xi} = \llbracket \lambda y.\, y \approx (\lambda x.\, \top)\rrbracket_{\mathcal{I}}^{\xi}$ and $\llbracket \exists\langle\tau\rangle\rrbracket_{\mathcal{I}}^{\xi} = \llbracket \lambda y.\, y \not\approx (\lambda x.\, \bot)\rrbracket_{\mathcal{I}}^{\xi}$ for all types τ and all valuations ξ. The argument we use here resembles the proof of Lemma 28, but here we cannot assume $\mathcal{I}$ to be proper.

Let f be a function from $\llbracket \tau\rrbracket_{\mathcal{I}_{\mathsf{ty}}}^{\xi}$ to $\{0,1\}$. Then

$$\llbracket \forall\langle\tau\rangle\rrbracket_{\mathcal{I}}^{\xi}(f) = \mathcal{J}(\forall, \llbracket \tau\rrbracket_{\mathcal{I}_{\mathsf{ty}}}^{\xi})(f) = \min\{f(a) \mid a \in \llbracket \tau\rrbracket_{\mathcal{I}_{\mathsf{ty}}}^{\xi}\} = \begin{cases}1 & \text{if } f \text{ is constantly } 1\\ 0 & \text{otherwise}\end{cases}$$

By our assumption on $\mathcal{I}$, we have

$$\begin{aligned}\llbracket \lambda y.\, y \approx (\lambda x.\, \top)\rrbracket_{\mathcal{I}}^{\xi}(f) &= \llbracket (y \approx (\lambda x.\, \top))\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi[y\mapsto f]} = \llbracket y \approx (\lambda x.\, \top)\rrbracket_{\mathcal{I}}^{\xi[y\mapsto f]}\\ &= \begin{cases}1 & \text{if } \llbracket y\rrbracket_{\mathcal{I}}^{\xi[y\mapsto f]} = \llbracket \lambda x.\, \top\rrbracket_{\mathcal{I}}^{\xi[y\mapsto f]}\\ 0 & \text{otherwise}\end{cases}\end{aligned}$$

Thus it remains to show that $\llbracket y\rrbracket_{\mathcal{I}}^{\xi[y\mapsto f]} = \llbracket \lambda x.\, \top\rrbracket_{\mathcal{I}}^{\xi[y\mapsto f]}$ if and only if f is constantly 1. This holds because $\llbracket y\rrbracket_{\mathcal{I}}^{\xi[y\mapsto f]}(a) = f(a)$ and, by our assumption on $\mathcal{I}$, $\llbracket \lambda x.\, \top\rrbracket_{\mathcal{I}}^{\xi[y\mapsto f]}(a) = \llbracket \top\rrbracket_{\mathcal{I}}^{\xi[x\mapsto a,\, y\mapsto f]} = 1$ for all $a \in \llbracket \tau\rrbracket_{\mathcal{I}_{\mathsf{ty}}}^{\xi}$.

The case of $\exists$ is analogous. □

Corollary 54. *Let $\mathcal{I}$ be an interpretation such that $\llbracket \lambda x.\, t\rrbracket_{\mathcal{I}}^{\xi}(a) = \llbracket t\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}}^{\xi[x\mapsto a]}$ for all λ-terms t and all valuations ξ. Then $\mathcal{I}$ is proper.*

Thus, to show that $\mathcal{I}^{\mathrm{GH}}$ is proper, it suffices to prove that $\llbracket \lambda x.\, t\rrbracket_{\mathcal{I}^{\mathrm{GH}}}^{\xi}(a) = \llbracket t\downarrow_{\mathsf{Q}_\approx}\rrbracket_{\mathcal{I}^{\mathrm{GH}}}^{\xi[x\mapsto a]}$. For quantifiers, we have the following relation between the higher-order interpretation $\mathcal{I}^{\mathrm{GH}}$ and the first-order interpretation R:

Lemma 55. *Let $\mathsf{Q} \in \{\forall, \exists\}$ and $f \in \mathcal{J}_{\mathsf{ty}}^{\mathrm{GH}}(\rightarrow)(\mathcal{D}_\tau, \{0,1\})$. Then, for any term p such that $\mathsf{Q}\langle\tau\rangle(\lambda x.\, p)$ is $\mathsf{Q}_\approx$-normal and such that $f = \mathcal{E}(\llbracket \mathcal{F}(\lambda x.\, p)\rrbracket_R)$,*

$$\mathcal{J}^{\mathrm{GH}}(\mathsf{Q}, \mathcal{D}_\tau)(f) = \mathcal{E}(\llbracket \mathcal{F}(\mathsf{Q}\langle\tau\rangle(\lambda x.\, p))\rrbracket_R)$$

Proof. Let Q, f, and p be as in the preconditions of the lemma. If $\mathsf{Q} = \forall$, then

$$\begin{aligned}
\mathcal{J}^{\mathrm{GH}}(\forall, \mathcal{D}_\tau)(f) &= \min\{f(a) \mid a \in \mathcal{D}_\tau\} \\
&= \min\{\mathcal{E}_{\tau\to\mathrm{o}}([\![\mathcal{F}(\lambda x.\, p)]\!]_R)(\mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R)) \mid \mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R) \in \mathcal{D}_\tau\} \\
&= \min\{\mathcal{E}_\mathrm{o}([\![\mathcal{F}((\lambda x.\, p)\, u)]\!]_R) \mid \mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R) \in \mathcal{D}_\tau\} \\
&= \min\{[\![\mathcal{F}((\lambda x.\, p)\, u)]\!]_R \mid [\![\mathcal{F}(u)]\!]_R \in \mathcal{U}_\tau\} \\
&= \min\{[\![\mathcal{F}(p\{x \mapsto u\})]\!]_R \mid [\![\mathcal{F}(u)]\!]_R \in \mathcal{U}_\tau\} \\
&= \min\{[\![\mathcal{F}(p)\{x \mapsto \mathcal{F}(u)\}]\!]_R \mid [\![\mathcal{F}(u)]\!]_R \in \mathcal{U}_\tau\} \\
&\qquad \text{since, by Lemma 7, } x \text{ never occurs under a } \lambda \text{ in } p, \\
&= \min\{[\![\mathcal{F}(p)]\!]_R^{\xi\{x\mapsto a'\}} \mid a' \in \mathcal{U}_\tau\} \\
&\qquad \text{by Lemma 6 (substitution lemma) of oSup,} \\
&= [\![\forall x.\, \mathcal{F}(p)]\!]_R = \mathcal{E}_\mathrm{o}([\![\mathcal{F}(\forall\langle\tau\rangle\, (\lambda x.\, p))]\!]_R)
\end{aligned}$$

If $\mathsf{Q} = \exists$, the proof is analogous, but uses max instead of min. □

A similar relation holds on all $\mathsf{Q}_\approx$-normal terms:

Lemma 56. *Given a ground $\mathsf{Q}_\approx$-normal λ-term t, we have*

$$[\![t]\!]_{\mathcal{J}^{\mathrm{GH}}} = \mathcal{E}([\![\mathcal{F}(t{\downarrow}_{\beta\eta\mathsf{Q}_\eta})]\!]_R)$$

Proof. The proof is analogous to that of Lemma 40 of λSup, using the $\beta\eta\mathsf{Q}_\eta$-normal form instead of the $\beta\eta$-normal form and with a special case for quantifier-headed terms. We proceed by induction on t. Assume that $[\![s]\!]_{\mathcal{J}^{\mathrm{GH}}} = \mathcal{E}([\![\mathcal{F}(s{\downarrow}_{\beta\eta\mathsf{Q}_\eta})]\!]_R)$ for all proper subterms s of t. If t is of the form $\mathsf{f}\langle\bar{\tau}\rangle$, then it cannot be a quantifier since t is $\mathsf{Q}_\approx$-normal. Thus:

$$\begin{aligned}
[\![t]\!]_{\mathcal{J}^{\mathrm{GH}}} &= \mathcal{J}^{\mathrm{GH}}(\mathsf{f}, \mathcal{D}_{\bar{\tau}}) \\
&= \mathcal{E}(\mathcal{J}(\mathsf{f}_0, \mathcal{U}_{\mathcal{F}(\bar{\tau})})) \\
&= \mathcal{E}([\![\mathsf{f}_0\langle\mathcal{F}(\bar{\tau})\rangle]\!]_R) \\
&= \mathcal{E}([\![\mathcal{F}(\mathsf{f}\langle\bar{\tau}\rangle)]\!]_R) \\
&= \mathcal{E}([\![\mathcal{F}(\mathsf{f}\langle\bar{\tau}\rangle{\downarrow}_{\beta\eta\mathsf{Q}_\eta})]\!]_R) = \mathcal{E}([\![\mathcal{F}(t{\downarrow}_{\beta\eta\mathsf{Q}_\eta})]\!]_R)
\end{aligned}$$

If t is of the form $t = \mathsf{Q}\langle\tau\rangle\, u$, then $u = \lambda x.\, v$ since t is $\mathsf{Q}_\approx$-normal, and

$$\begin{aligned}
[\![\mathsf{Q}\langle\tau\rangle\, (\lambda x.\, v)]\!]_{\mathcal{J}^{\mathrm{GH}}} &= [\![\mathsf{Q}\langle\tau\rangle]\!]_{\mathcal{J}^{\mathrm{GH}}}\, [\![\lambda x.\, v]\!]_{\mathcal{J}^{\mathrm{GH}}} \\
&= \mathcal{E}_\mathrm{o}([\![\mathcal{F}(\mathsf{Q}\langle\tau\rangle\, (\lambda x.\, v))]\!]_R) && \text{by Lemma 55} \\
&= \mathcal{E}_\mathrm{o}([\![\mathcal{F}((\mathsf{Q}\langle\tau\rangle\, (\lambda x.\, v)){\downarrow}_{\beta\eta\mathsf{Q}_\eta})]\!]_R) && \text{by the definition of } \mathcal{F}
\end{aligned}$$

If t is an application $t = t_1\, t_2$, where t_1 is of type $\tau \to \upsilon$ and t_1 is not a quantifier, then

$$\begin{aligned}
[\![t_1\, t_2]\!]_{\mathcal{J}^{\mathrm{GH}}} &= [\![t_1]\!]_{\mathcal{J}^{\mathrm{GH}}}([\![t_2]\!]_{\mathcal{J}^{\mathrm{GH}}}) \\
&\overset{\mathrm{IH}}{=} \mathcal{E}_{\tau\to\upsilon}([\![\mathcal{F}(t_1{\downarrow}_{\beta\eta\mathsf{Q}_\eta})]\!]_R)(\mathcal{E}_\tau([\![\mathcal{F}(t_2{\downarrow}_{\beta\eta\mathsf{Q}_\eta})]\!]_R)) \\
&= \mathcal{E}_\upsilon([\![\mathcal{F}((t_1\, t_2){\downarrow}_{\beta\eta\mathsf{Q}_\eta})]\!]_R)
\end{aligned}$$

If t is a λ-expression, then

$$\begin{aligned}
[\![\lambda x.\, u]\!]^{\xi}_{\mathcal{I}^{\mathrm{GH}}} &= \mathcal{L}^{\mathrm{GH}}(\xi, (\lambda x.\, u)) \\
&= \mathcal{E}([\![\mathcal{F}((\lambda x.\, u)\theta{\downarrow_{\beta\eta\mathsf{Q}_\eta}})]\!]_R) \\
&= \mathcal{E}([\![\mathcal{F}((\lambda x.\, u){\downarrow_{\beta\eta\mathsf{Q}_\eta}})]\!]_R)
\end{aligned}$$

where θ is a substitution such that $\mathcal{D}_{\alpha\theta} = \xi(\alpha)$ and $\mathcal{E}([\![\mathcal{F}(x\theta)]\!]_R) = \xi(x)$. $\square$

We also need to employ the following lemma, which is very similar to the substitution lemma, but we must prove it here for our particular interpretation $\mathcal{I}^{\mathrm{GH}}$ because we have not shown that $\mathcal{I}^{\mathrm{GH}}$ is proper yet.

Lemma 57 (Substitution lemma). *$[\![\tau\rho]\!]^{\xi}_{\mathcal{I}^{\mathrm{GH}}_{\mathsf{ty}}} = [\![\tau]\!]^{\xi'}_{\mathcal{I}^{\mathrm{GH}}_{\mathsf{ty}}}$ and $[\![t\rho]\!]^{\xi}_{\mathcal{I}^{\mathrm{GH}}} = [\![t]\!]^{\xi'}_{\mathcal{I}^{\mathrm{GH}}}$ for all $\mathsf{Q}_{\approx}$-normal λ-terms t, all $\tau \in \mathcal{T}y_{\mathrm{H}}$ and all grounding substitutions ρ, where $\xi'(\alpha) = [\![\alpha\rho]\!]^{\xi}_{\mathcal{I}^{\mathrm{GH}}_{\mathsf{ty}}}$ for all type variables α and $\xi'(x) = [\![x\rho]\!]^{\xi}_{\mathcal{I}^{\mathrm{GH}}}$ for all term variables x.*

Proof. Analogous to Lemma 41 of λSup, using the $\beta\eta\mathsf{Q}_\eta$-normal form instead of the $\beta\eta$-normal form.

Lemma 58. *The interpretation $\mathcal{I}^{\mathrm{GH}}$ is proper.*

Proof. By Corollary 54, it is enough to show that $[\![\lambda x.\, t]\!]^{\xi}_{\mathcal{I}^{\mathrm{GH}}}(a) = [\![t\downarrow_{\mathsf{Q}_{\approx}}]\!]^{\xi[x\mapsto a]}_{\mathcal{I}^{\mathrm{GH}}}$. First, we show it for all $\mathsf{Q}_{\approx}$-normal λ-expressions $\lambda x.\, t$, all valuations ξ, and all values a:

$$\begin{aligned}
[\![\lambda x.\, t]\!]^{\xi}_{\mathcal{I}^{\mathrm{GH}}}(a) &= \mathcal{L}^{\mathrm{GH}}(\xi, \lambda x.\, t)(a) && \text{by the definition of } [\![\ ]\!]_{\mathcal{I}^{\mathrm{GH}}} \\
&= \mathcal{E}([\![\mathcal{F}((\lambda x.\, t)\theta{\downarrow_{\beta\eta\mathsf{Q}_\eta}})]\!]_R)(a) && \text{by the definition of } \mathcal{L}^{\mathrm{GH}} \text{ for some } \theta \\
& && \text{such that } \mathcal{E}([\![\mathcal{F}(z\theta)]\!]_R) = \xi(z) \text{ for all } z \\
& && \text{and } \mathcal{D}_{\alpha\theta} = \xi(\alpha) \text{ for all } \alpha \\
&= \mathcal{E}([\![\mathcal{F}(((\lambda x.\, t)\theta\ s){\downarrow_{\beta\eta\mathsf{Q}_\eta}})]\!]_R) && \text{by the definition of } \mathcal{E} \\
& && \text{where } \mathcal{E}([\![\mathcal{F}(s)]\!]_R) = a \\
&= \mathcal{E}([\![\mathcal{F}(t(\theta[x\mapsto s]){\downarrow_{\beta\eta\mathsf{Q}_\eta}})]\!]_R) && \text{by } \beta\text{-reduction} \\
&= [\![t(\theta[x\mapsto s])]\!]_{\mathcal{I}^{\mathrm{GH}}} && \text{by Lemma 56} \\
&= [\![t]\!]^{\xi[x\mapsto a]}_{\mathcal{I}^{\mathrm{GH}}} && \text{by Lemma 57} \\
&= [\![t\downarrow_{\mathsf{Q}_{\approx}}]\!]^{\xi[x\mapsto a]}_{\mathcal{I}^{\mathrm{GH}}} && \text{because } t \text{ is } \mathsf{Q}_{\approx}\text{-normal by definition}
\end{aligned}$$

The case where $\lambda x.\, t$ is not $\mathsf{Q}_{\approx}$-normal is reduced to the previous case because then $[\![\lambda x.\, t]\!]^{\xi}_{\mathcal{I}^{\mathrm{GH}}}(a) = \mathcal{L}^{\mathrm{GH}}(\xi, (\lambda x.\, t)\downarrow_{\mathsf{Q}_{\approx}})(a)$ and $(\lambda x.\, t)\downarrow_{\mathsf{Q}_{\approx}} = \lambda x. t'$ where $t' = t\downarrow_{\mathsf{Q}_{\approx}}$ by definition. $\square$

Lemma 59. *$\mathcal{I}^{\mathrm{GH}}$ is a model of N.*

Proof. Because all terms in N are $\mathsf{Q}_{\approx}$-normal, by Lemma 56, $[\![t]\!]_{\mathcal{I}^{\mathrm{GH}}} = \mathcal{E}([\![\mathcal{F}(t)]\!]_R)$ for all $t \in \mathcal{T}_{\mathrm{GH}}$. Since $\mathcal{E}$ is a bijection, it follows that any literal $s \approx t \in \mathcal{C}_{\mathrm{GH}}$ is true in $\mathcal{I}^{\mathrm{GH}}$ if and only if $\mathcal{F}(s \approx t)$ is true in R. Hence, a clause $C \in \mathcal{C}_{\mathrm{GH}}$ is true in $\mathcal{I}^{\mathrm{GH}}$ if and only if $\mathcal{F}(C)$ is true in R. By Theorem 50 and the assumption that $\bot \notin N$, R is a model of $\mathcal{F}(N)$—that is, for all clauses $C \in N$, $\mathcal{F}(C)$ is true in R. Hence, all clauses $C \in N$ are true in $\mathcal{I}^{\mathrm{GH}}$ and therefore $\mathcal{I}^{\mathrm{GH}}$ is a model of N. $\square$

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

Theorem 60 (Ground static completeness). *Let $q \in Q$ be some parameter triple. Then $GHInf^q$ is statically refutationally complete w.r.t. $\models$ and $(GHRed_{\mathrm{I}}^q, GHRed_{\mathrm{C}})$. In other words, if $N \subseteq \mathcal{C}_{\mathrm{GH}}$ is a clause set saturated w.r.t. $GHInf^q$ and $GHRed_{\mathrm{I}}^q$, then $N \models \bot$ if and only if $\bot \in N$.*

The construction of $\mathcal{I}^{\mathrm{GH}}$ relies on the specific properties of R. It would not work with an arbitrary interpretation. In the other direction, transforming a higher-order model into a first-order model with interpreted Booleans is easier, as the following lemma shows:

Lemma 61. *Given a proper higher-order 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 $\mathcal{I} = (\mathcal{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$ be a proper higher-order interpretation on GH. Let $\mathcal{U}_\tau^{\mathrm{GF}} = [\![\tau]\!]_{\mathcal{I}_{\mathsf{ty}}}$ be the GF universe for the ground type τ. For a symbol $\mathsf{f}_j^{\bar{\upsilon}} \in \Sigma_{\mathrm{GF}}$, let $\mathcal{J}^{\mathrm{GF}}(\mathsf{f}_j^{\bar{\upsilon}}) = [\![\mathsf{f}\langle\bar{\upsilon}\rangle]\!]_{\mathcal{I}}$ (up to currying). For a symbol $\mathsf{lam}_{\lambda x.\, t} \in \Sigma_{\mathrm{GF}}$, let $\mathcal{J}^{\mathrm{GF}}(\mathsf{lam}_{\lambda x.\, t}) = [\![\lambda x.\, t]\!]_{\mathcal{I}}$.

The requirements on the GF-interpretation of logical symbols are fulfilled because we have similar requirements on H: $\mathcal{U}_{\mathrm{o}}^{\mathrm{GF}} = \mathcal{J}_{\mathsf{ty}}(\mathrm{o}) = \{0,1\}$; $\mathcal{J}^{\mathrm{GF}}(\top_0) = \mathcal{J}(\top) = 1$; $\mathcal{J}^{\mathrm{GF}}(\neg_1)(a) = \mathcal{J}(\neg)(a) = 1 - a$; and similarly for the other logical symbols. Thus, 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 $[\![t]\!]_{\mathcal{I}}^{\xi} = [\![\mathcal{F}(t)]\!]_{\mathcal{I}^{\mathrm{GF}}}^{\xi}$ for all terms $t \in \mathcal{T}_{\mathrm{GH}}$. We prove this by induction on the structure of the $\beta\eta\mathsf{Q}_\eta$-normal form of t. If t is a λ-expression, this is obvious. If t is of the form $\mathsf{f}\langle\bar{\upsilon}\rangle\,\bar{s}_j$, then $\mathcal{F}(t) = \mathsf{f}_j^{\bar{\upsilon}}(\mathcal{F}(\bar{s}_j))$ and hence

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

If t is of the form $\forall\langle\tau\rangle\,(\lambda x.\, s)$, then $\mathcal{F}(t) = \forall x.\, \mathcal{F}(s)$ and hence

$$\begin{aligned}[\![\mathcal{F}(t)]\!]_{\mathcal{I}^{\mathrm{GF}}}^{\xi} &= \min\{[\![\mathcal{F}(s)]\!]_{\mathcal{I}^{\mathrm{GF}}}^{\xi[x\mapsto a]} \mid a \in \mathcal{U}_\tau^{\mathrm{GF}}\} \overset{\mathrm{IH}}{=} \min\{[\![s]\!]_{\mathcal{I}}^{\xi[x\mapsto a]} \mid a \in [\![\tau]\!]_{\mathcal{I}_{\mathsf{ty}}}\} \\ &= \min\{[\![\lambda x.\, s]\!]_{\mathcal{I}}^{\xi}(a) \mid a \in [\![\tau]\!]_{\mathcal{I}_{\mathsf{ty}}}\} = [\![t]\!]_{\mathcal{I}}^{\xi}\end{aligned}$$

A similar argument applies for $\exists$. Since the definition of $\mathcal{F}$ recurses into subterms below quantifiers, we finally need to consider the case where t is a variable x. In that case, we have $\mathcal{F}(x) = x$ and hence $[\![\mathcal{F}(t)]\!]_{\mathcal{I}^{\mathrm{GF}}}^{\xi} = \xi(x) = [\![t]\!]_{\mathcal{I}}^{\xi}$. □

4.4 The Nonground Higher-Order Level

To lift the result to the nonground level, we employ the saturation framework of Waldmann et al. [35]. Clearly, the entailment relation $\models$ on GH qualifies as a consequence relation in the sense of the framework. We need to show that our redundancy criterion on GH qualifies as a redundancy criterion and that $\mathcal{G}$ qualifies as a grounding function:

Lemma 62. *The pair $(GHRed_{\mathrm{I}}^q, GHRed_{\mathrm{C}})$ is a redundancy criterion in the sense of the saturation framework.*

Proof. We must prove the conditions (R1) to (R4) of the saturation framework. Adapted to our context, they state the following for all clause sets $N, N' \subseteq \mathcal{C}_{\mathrm{GH}}$:

(R1) if $N \models \bot$, then $N \setminus \mathit{GHRed}_{\mathrm{C}}(N) \models \bot$;
(R2) if $N \subseteq N'$, then $\mathit{GHRed}_{\mathrm{C}}(N) \subseteq \mathit{GHRed}_{\mathrm{C}}(N')$ and $\mathit{GHRed}_{\mathrm{I}}(N) \subseteq \mathit{GHRed}_{\mathrm{I}}(N')$;
(R3) if $N' \subseteq \mathit{GHRed}_{\mathrm{C}}(N)$, then $\mathit{GHRed}_{\mathrm{C}}(N) \subseteq \mathit{GHRed}_{\mathrm{C}}(N \setminus N')$ and $\mathit{GHRed}_{\mathrm{I}}(N) \subseteq \mathit{GHRed}_{\mathrm{I}}(N \setminus N')$;
(R4) if $\iota \in \mathit{GHInf}$ and $\mathit{concl}(\iota) \in N$, then $\iota \in \mathit{GHRed}_{\mathrm{I}}(N)$.

For (R1), it suffices to show that $N \setminus \mathit{GHRed}_{\mathrm{C}}(N) \models N$. Let $\mathcal{I}$ be a model of $N \setminus \mathit{GHRed}_{\mathrm{C}}(N)$. By Lemma 61, there exists a model $\mathcal{I}^{\mathrm{GF}}$ of $\mathcal{F}(N \setminus \mathit{GHRed}_{\mathrm{C}}(N)) = \mathcal{F}(N) \setminus \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N))$. We show that $\mathcal{I}^{\mathrm{GF}} \models C$ for each clause $C \in \mathcal{F}(N)$ by well-founded induction on C w.r.t. $\succ$. If $C \notin \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N))$, we have already shown that $\mathcal{I}^{\mathrm{GF}} \models C$. Otherwise, $C \in \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N))$ and hence $\{D \in \mathcal{F}(N) \mid D \prec C\} \models C$. By the induction hypothesis, it follows that $\mathcal{I}^{\mathrm{GF}} \models C$. Thus, we have shown that $\mathcal{I}^{\mathrm{GF}} \models \mathcal{F}(N)$. By Lemma 61, this implies $\mathcal{I} \models N$.

For the first part of (R2), let $N \subseteq N'$ and $C \in \mathit{GHRed}_{\mathrm{C}}(N)$, i.e., $\{D \in \mathcal{F}(N) \mid D \prec \mathcal{F}(C)\} \models \mathcal{F}(C)$. We must show that $\{D \in \mathcal{F}(N') \mid D \prec \mathcal{F}(C)\} \models \mathcal{F}(C)$. This is obvious because $\{D \in \mathcal{F}(N) \mid D \prec \mathcal{F}(C)\} \subseteq \{D \in \mathcal{F}(N') \mid D \prec \mathcal{F}(C)\}$.

For the second part of (R2), let $N \subseteq N'$ and $\iota \in \mathit{GHRed}_{\mathrm{I}}(N)$. We must show that $\iota \in \mathit{GHRed}_{\mathrm{I}}(N')$. If ι is a GArgCong, GExt, or GChoice inference, we have $\mathit{concl}(\iota) \in N \cup \mathit{GHRed}_{\mathrm{C}}(N)$. Using the first part of (R2), it follows that $N \cup \mathit{GHRed}_{\mathrm{C}}(N) \subseteq N' \cup \mathit{GHRed}_{\mathrm{C}}(N')$, which implies $\iota \in \mathit{GHRed}_{\mathrm{I}}(N')$. If ι is some other kind of inference, we have $\mathit{prems}(\mathcal{F}(\iota)) \cap \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N)) \neq \varnothing$ or $\{D \in \mathcal{F}(N) \mid D \prec \mathit{mprem}(\mathcal{F}(\iota))\} \models \mathit{concl}(\mathcal{F}(\iota))$. In the first case, $\mathit{prems}(\mathcal{F}(\iota)) \cap \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N')) \neq \varnothing$ because by the first part of (R2), we have $\mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N)) \subseteq \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N'))$. In the second case, we have $\{D \in \mathcal{F}(N') \mid D \prec \mathit{mprem}(\mathcal{F}(\iota))\} \models \mathit{concl}(\mathcal{F}(\iota))$ because $N \subseteq N'$ implies $\{D \in \mathcal{F}(N) \mid D \prec \mathit{mprem}(\mathcal{F}(\iota))\} \subseteq \{D \in \mathcal{F}(N') \mid D \prec \mathit{mprem}(\mathcal{F}(\iota))\}$.

For the first part of (R3), let $N' \subseteq \mathit{GHRed}_{\mathrm{C}}(N)$ and $C \in \mathit{GHRed}_{\mathrm{C}}(N)$, i.e., $\{D \in \mathcal{F}(N) \mid D \prec \mathcal{F}(C)\} \models \mathcal{F}(C)$. We must show that $\{D \in \mathcal{F}(N \setminus N') \mid D \prec \mathcal{F}(C)\} \models \mathcal{F}(C)$. Let $\mathcal{I}$ be a model of $\{D \in \mathcal{F}(N \setminus N') \mid D \prec \mathcal{F}(C)\}$. It suffices to show that $\mathcal{I} \models \{D \in \mathcal{F}(N) \mid D \prec \mathcal{F}(C)\}$, meaning $\mathcal{I} \models E$ for every $E \in \mathcal{F}(N)$ such that $E \prec \mathcal{F}(C)$. We prove this by well-founded induction on E w.r.t. $\succ$. If $E \in \mathcal{F}(N \setminus N')$, the claim holds by assumption. Otherwise, $E \in \mathcal{F}(N') \subseteq \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N))$; hence $\{D \in \mathcal{F}(N) \mid D \prec E\} \models E$ and therefore $\mathcal{I} \models E$ by the induction hypothesis.

For the second part of (R3), let $N' \subseteq \mathit{GHRed}_{\mathrm{C}}(N)$ and $\iota \in \mathit{GHRed}_{\mathrm{I}}(N)$. We must show that $\iota \in \mathit{GHRed}_{\mathrm{I}}(N \setminus N')$. If ι is a GArgCong, GExt, or GChoice inference, we have $\mathit{concl}(\iota) \in N \cup \mathit{GHRed}_{\mathrm{C}}(N)$. Using $N' \subseteq \mathit{GHRed}_{\mathrm{C}}(N)$, and by the first part of (R3), it follows that $\mathit{concl}(\iota) \in N \cup \mathit{GHRed}_{\mathrm{C}}(N) = (N \setminus N') \cup \mathit{GHRed}_{\mathrm{C}}(N) \subseteq (N \setminus N') \cup \mathit{GHRed}_{\mathrm{C}}(N \setminus N')$ and therefore $\iota \in \mathit{GHRed}_{\mathrm{I}}(N \setminus N')$. If ι is some other kind of inference, we have $\mathit{prems}(\mathcal{F}(\iota)) \cap \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N)) \neq \varnothing$ or $\{D \in \mathcal{F}(N) \mid D \prec \mathit{mprem}(\mathcal{F}(\iota))\} \models \mathit{concl}(\mathcal{F}(\iota))$. In the first case, $\mathit{prems}(\mathcal{F}(\iota)) \cap \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N \setminus N')) \neq \varnothing$ because by the first part of (R3), we have $\mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N)) \subseteq \mathit{GFRed}_{\mathrm{C}}(\mathcal{F}(N \setminus N'))$. In the second case, it suffices to show that $\{D \in \mathcal{F}(N \setminus N') \mid D \prec \mathit{mprem}(\mathcal{F}(\iota))\} \models \{D \in \mathcal{F}(N) \mid D \prec \mathit{mprem}(\mathcal{F}(\iota))\}$, which can be shown analogously to the induction used for the first part of (R3).

For (R4), let $\iota \in \mathit{GHInf}$ and $\mathit{concl}(\iota) \in N$. We must show that $\iota \in \mathit{GHRed}_{\mathrm{I}}(N)$. If ι is a GARGCONG, GEXT, or GCHOICE inference, we must show $\mathit{concl}(\iota) \in N \cup \mathit{GHRed}_{\mathrm{C}}(N)$, which obviously holds by assumption. If ι is some other kind of inference, it suffices to show $\{D \in \mathcal{F}(N) \mid D \prec \mathit{mprem}(\mathcal{F}(\iota))\} \models \mathit{concl}(\mathcal{F}(\iota))$. This holds because $\mathit{concl}(\mathcal{F}(\iota)) \in \mathcal{F}(N)$ and $\mathit{concl}(\mathcal{F}(\iota)) \prec \mathit{mprem}(\mathcal{F}(\iota))$. □

Lemma 63. *For every $q \in Q$, the function $\mathcal{G}^q$ is a grounding function in the sense of the saturation framework.*

Proof. We must prove the conditions (G1), (G2), and (G3) of the saturation framework. Adapted to our context, they state the following:

(G1) $\mathcal{G}(\bot) = \{\bot\}$;
(G2) for every $C \in \mathcal{C}_{\mathrm{H}}$, if $\bot \in \mathcal{G}(C)$, then $C = \bot$;
(G3) for every $\iota \in \mathit{HInf}$, $\mathcal{G}^q(\iota) \subseteq \mathit{GHRed}^q_{\mathrm{I}}(\mathcal{G}(\mathit{concl}(\iota)))$.

Clearly, $C = \bot$ if and only if $\bot \in \mathcal{G}(C)$ if and only if $\mathcal{G}(C) = \{\bot\}$, proving (G1) and (G2). For every $\iota \in \mathit{HInf}$, by the definition of $\mathcal{G}^q$ (Definition 42) and by Lemma 39, we have $\mathit{concl}(\mathcal{G}^q(\iota)) \subseteq \mathcal{G}(\mathit{concl}(\iota))$, and thus (G3) by (R4). □

To lift the completeness result of the previous subsection to the nonground calculus HInf, we employ Theorem 14 of the saturation framework, which, adapted to our context, is stated as follows.

Theorem 64 (Lifting theorem). *If GHInf^q is statically refutationally complete w.r.t. $(\mathit{GHRed}^q_{\mathrm{I}}, \mathit{GHRed}_{\mathrm{C}})$ for every parameter triple $q \in Q$, and if for every $N \subseteq \mathcal{C}_{\mathrm{H}}$ that is saturated w.r.t. HInf and $\mathit{HRed}_{\mathrm{I}}$ there exists a $q \in Q$ such that $\mathit{GHInf}^q(\mathcal{G}(N)) \subseteq \mathcal{G}^q(\mathit{HInf}(N)) \cup \mathit{GHRed}^q_{\mathrm{I}}(\mathcal{G}(N))$, then also HInf is statically refutationally complete w.r.t. $(\mathit{HRed}_{\mathrm{I}}, \mathit{HRed}_{\mathrm{C}})$ and $\models_{\mathcal{G}}$.*

Proof. This is almost an instance of Theorem 14 of the saturation framework. We take $\mathcal{C}_{\mathrm{H}}$ for **F**, $\mathcal{C}_{\mathrm{GH}}$ for **G**. Clearly, the entailment relation $\models$ on GH is a consequence relation in the sense of the framework. By Lemma 62 and 63, $(\mathit{GHRed}^q_{\mathrm{I}}, \mathit{GHRed}_{\mathrm{C}})$ is a redundancy criterion in the sense of the framework, and $\mathcal{G}^q$ are grounding functions in the sense of the framework, for all $q \in Q$. The redundancy criterion $(\mathit{HRed}_{\mathrm{I}}, \mathit{HRed}_{\mathrm{C}})$ matches exactly the intersected lifted redundancy criterion $\mathit{Red}^{\cap\mathcal{G},\sqsupset}$ of the saturation framework. Their Theorem 14 states the theorem only for $\sqsupset = \varnothing$. By their Lemma 16, it also holds if $\sqsupset \neq \varnothing$. □

Let $N \subseteq \mathcal{C}_{\mathrm{H}}$ be a clause set saturated w.r.t. HInf and $\mathit{HRed}_{\mathrm{I}}$. For the above theorem to apply, we need to show that there exists a $q \in Q$ such that all inferences $\iota \in \mathit{GHInf}^q$ with $\mathit{prems}(\iota) \in \mathcal{G}(N)$ are liftable or redundant. Here, ι being *liftable* means that ι is a $\mathcal{G}^q$-ground instance of an HInf-inference from N; ι being *redundant* means that $\iota \in \mathit{GHRed}^q_{\mathrm{I}}(\mathcal{G}(N))$.

To choose the right $q = (\mathit{GHLitSel}, \mathit{GHBoolSel}, \mathit{GHWit}) \in Q$, 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 we choose $\mathit{GHLitSel}$ and $\mathit{GHBoolSel}$ such that the selections in C correspond to those in $\mathcal{G}^{-1}(C)$.

To choose the witness function *GHWit*, let $C \in \mathcal{C}_{\mathrm{GH}}$ and let p a green position of a quantifier-headed term $C|_p = \mathsf{Q}\langle\tau\rangle\, t$. Let $D = \mathcal{G}^{-1}(C)$ and let θ be the grounding substitution such that $D\theta = C$. Let p' be the green position corresponding to p in D. If there exists no such position p', we define $\mathit{GHWit}(C, p)$ to be some arbitrary term that fulfills the order requirements of a witness function. Otherwise, let β and y be fresh variables and we extend θ to a substitution θ' by defining $\beta\theta' = \tau$ and $y\theta' = t$. Then θ' is a unifier of $\mathsf{Q}\langle\beta\rangle\, y$ and $D|_{p'}$ and hence there exists an idempotent $\sigma \in \mathrm{CSU}(Q\langle\beta\rangle\, y, D|_{p'})$ such that for some substitution ρ and for all variables x in D and for $x \in \{y, \beta\}$, we have $x\sigma\rho = x\theta'$. We let $\mathit{GHWit}(C, p)$ be $\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,\neg(y\sigma\, z)}\langle\bar{\alpha}\rangle\, \bar{x}\theta$ if the quantifier-headed term is a $\forall$-term and $\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,(y\sigma\, z)}\langle\bar{\alpha}\rangle\, \bar{x}\theta$ if the quantifier-headed term is an $\exists$-term where $\bar{\alpha}$ are the free type variables and $\bar{x}$ are the free variables occurring in $D|_{p'}$ in order of first occurrence.

By definition of $\mathcal{G}$ (Definition 32), for all variables x occurring in D the only Boolean green subterms of $x\theta$ are $\top$ and $\bot$. The term $\mathsf{Q}\langle\tau\rangle\, t$ must be $\mathsf{Q}_{\approx}$-normal because it occurs in $C \in \mathcal{C}_{\mathrm{GH}}$. Hence $\mathsf{Q}\langle\tau\rangle\, t \succ t\, \mathit{GHWit}(C, p)$ by order condition O4.

With respect to this parameter triple $q = (\mathit{GHLitSel}, \mathit{GHBoolSel}, \mathit{GHWit})$, we can show that all inferences from $\mathcal{G}(N)$ are liftable or redundant:

Lemma 65. *Let $C\theta \in \mathcal{C}_{\mathrm{GH}}$ and $C = \mathcal{G}^{-1}(C\theta)$. Let σ and ρ be substitutions such that $x\sigma\rho = x\theta$ for all variables in C. (This holds for example if σ is an element of a CSU corresponding to a unifier θ.) If a literal in a clause $C\theta$ is (strictly) $\succeq$-eligible w.r.t. GHLitSel, then a corresponding literal in C is (strictly) $\gtrsim$-eligible w.r.t. σ and HLitSel. If a green position in a clause $C\theta$ is $\succeq$-eligible w.r.t. GHBoolSel and there exists a corresponding green position in C, then the corresponding position in C is $\gtrsim$-eligible w.r.t. σ and HBoolSel.*

Proof. LITERALS: If the literal in $C\theta$ is selected w.r.t. *GHLitSel*, then the corresponding literal is also selected in $C = \mathcal{G}^{-1}(C\theta)$ w.r.t. *HLitSel* by definition of *GHLitSel*. If $L\theta$ is (strictly) $\succeq$-maximal in $C\theta$, then $L\sigma$ is (strictly) $\gtrsim$-maximal in $C\sigma$.

POSITIONS: Let p be the position in $C\theta$ and let p' be the corresponding position in C. We proceed by induction over the definition of eligible positions. If p is selected in $C\theta$ w.r.t. *GHBoolSel*, then p' is selected in $C = \mathcal{G}^{-1}(C\theta)$ w.r.t. *HBoolSel* by definition of *GHBoolSel*. Otherwise, if p is at the top level of a literal $L\theta = s\theta \approx t\theta$, then $s\theta \not\preceq t\theta$ implies $s\sigma \not\lesssim t\sigma$, (strict) $\succeq$-eligibility of $L\theta$ implies (strict) $\gtrsim$-eligibility of L w.r.t. σ (as shown above), and hence p' is eligible in C w.r.t. σ. Otherwise, the position p is neither selected nor at the top level. Let q be the position directly above p and q' be the position directly above p'. By the induction hypothesis, q and q' are eligible. If the head of $C\theta_p$ is not $\boldsymbol{\approx}$ or $\boldsymbol{\not\approx}$, then the head of $C_{p'}$ cannot be $\boldsymbol{\approx}$ or $\boldsymbol{\not\approx}$ either. If the head $C\theta_p$ is $\boldsymbol{\approx}$ or $\boldsymbol{\not\approx}$, then $C_{p'}$ must also be $\boldsymbol{\approx}$ or $\boldsymbol{\not\approx}$ because the position p' is green. Hence, p' is eligible because $s\theta \not\preceq t\theta$ implies $s\sigma \not\lesssim t\sigma$. $\square$

In some edge cases, it is ambiguous what "the corresponding" literal is. When $C\theta$ contains multiple occurrences of a literal that correspond to different literals in C, the $\gtrsim$-larger one must be chosen as the corresponding literal to make the lemma above work. In the following, we will implicitly assume that the correct literal is chosen when we refer to "the corresponding" literal.

Lemma 66. *All* ERES, EFACT, GARGCONG, GEXT, GCHOICE, BOOLHOIST, *and* FALSEELIM *inferences are liftable.*

Proof. For ERES, EFACT, GARGCONG, and GEXT, the proof is as in Lemma 50 of λSup [7]. For GCHOICE, the proof is analogous to GEXT.

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

$$\frac{C\theta\langle u\rangle_p}{C\theta\langle\bot\rangle_p \vee u \approx \top}\ \text{BOOLHOIST}$$

where $\mathcal{G}^{-1}(C\theta) = C$.

If p corresponds to a position at or below an unapplied variable in C, u could only be $\top$ or $\bot$, contradicting the condition of BOOLHOIST that u is not a fully applied logical symbol.

If p corresponds to a position at or below a fluid term in C, we will lift to a FLUIDBOOLHOIST inference. Let $p = p_1.p_2$ such that p_1 is the longest prefix of p that corresponds to a green position p_1' in C. Let $v = C|_{p_1'}$. Then v is fluid. Let z and x be fresh variables. Define a substitution θ' that maps the variable z to $\lambda y.(v\theta)\langle y\rangle_{p_2}$, the variable x to $v\theta|_{p_2}$, and all other variables w to $w\theta$. Then $(z\,x)\theta' = (v\theta)\langle v\theta|_{p_2}\rangle_{p_2} = v\theta = v\theta'$. So θ' is a unifier of $z\,x$ and v and thus there exists an idempotent $\sigma \in \mathrm{CSU}(z\,x, v)$ such that for some substitution ρ, for all variables y in C, and for $y \in \{x, z\}$, we have $y\sigma\rho = y\theta'$. By the conditions of BOOLHOIST, $u \neq \top$ and $u \neq \bot$. Then $x\sigma \neq \top$ and $x\sigma \neq \bot$ because $u = v\theta|_{p_2} = x\theta' = x\sigma\rho$. Hence, we have $(z\,\bot)\theta' = (v\theta)\langle\bot\rangle_{p_2} \neq (v\theta)\langle x\theta'\rangle_{p_2} = (z\,x)\theta'$ and thus $(z\,\bot)\sigma \neq (z\,x)\sigma$. The position p_1 must be eligible in $C\theta$ because p is eligible in $C\theta$ and p_1 is the longest prefix of p that corresponds to a green position p_1' in C. Eligibility of p_1 in $C\theta$ implies eligibility of p_1' in C by Lemma 65. Thus there exists the following FLUIDBOOLHOIST inference ι':

$$\frac{C\langle v\rangle_{p_1'}}{(C\langle z\,\bot\rangle_{p_1'} \vee x \approx \top)\sigma}\ \text{FLUIDBOOLHOIST}$$

The inference ι is the $\sigma\rho$-ground instance of ι' and is therefore liftable.

Otherwise, we will lift to a BOOLHOIST inference. Since u is not at or below a variable-headed term, there is a subterm u' of C at position p' corresponding to the subterm u of $C\theta$ at position p. Since u is a Boolean term, there is a type unifier σ of the type of u' with the Boolean type. Eligibility of u in $C\theta$ implies eligibility of u' in C by Lemma 65. Since the occurrence of u in $C\theta$ is not at the top level of a positive literal, the corresponding occurrence of u' in C is not at the top level of a positive literal either. Thus there exists the following BOOLHOIST inference ι':

$$\frac{C\langle u'\rangle_{p'}}{(C\langle\bot\rangle_{p'} \vee u' \approx \top)\sigma}\ \text{BOOLHOIST}$$

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

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

$$\frac{C\theta \;=\; C'\theta \vee s\theta \approx s'\theta}{C'\theta}\,\textsc{FalseElim}$$

where $\mathcal{G}^{-1}(C\theta) = C = C' \vee s \approx s'$ and the literal $s\theta \approx s'\theta$ is strictly $\succeq$-eligible w.r.t. *GHLitSel*. Since $s\theta \approx s'\theta$ and $\bot \approx \top$ are unifiable and ground, we have $s\theta = \bot$ and $s'\theta = \top$. Thus, there exists an idempotent $\sigma \in \mathrm{CSU}(s \approx s', \bot \approx \top)$ such that for some substitution ρ and for all variables x in C, we have $x\sigma\rho = x\theta$. Then $s \approx s'$ is strictly $\gtrsim$-eligible in C w.r.t. σ. Hence, the following inference $\iota' \in \mathit{HInf}$ is applicable:

$$\frac{C' \vee s \approx s'}{C'\sigma}\,\textsc{FalseElim}$$

Then ι is the $\sigma\rho$-ground instance of ι' and is therefore liftable.

Lemma 67. *All* SUP *inferences are liftable or redundant.*

Proof. The proof is as for Lemmas 52 and 53 of λSup [7]. The proof works with the altered definition of deeply occurring variables because congruence holds below quantifiers on the GF level.

Lemma 68. *All* EQHOIST, NEQHOIST, GFORALLHOIST, GEXISTSHOIST, GFORALLRW, GEXISTSRW, *and* BOOLRW *inferences from $\mathcal{G}(N)$ are liftable or redundant.*

Proof. Let $\iota \in \mathit{GHInf}$ be a EQHOIST, NEQHOIST, GFORALLHOIST, GEXISTSHOIST, GFORALLRW, GEXISTSRW, or BOOLRW inference from $\mathcal{G}(N)$. Let $C\theta = \mathit{prems}(\iota)$ where $C = \mathcal{G}^{-1}(C\theta) \in N$. Let p be the position of the affected subterm in $C\theta$.

We distinguish two cases. We will show that ι is liftable if

(A) p corresponds to a position in C that is not at or below a fluid term, or
(B) p is the position of a term v in a literal $v \approx \top$ or $v \approx \bot$ in $C\theta$.

Otherwise, we will show that ι is redundant.

LIFTABLE CASES: If condition A or B holds, p corresponds to some position p' in C. Let $u = C|_{p'}$. By the definition of the grounding function $\mathcal{G}$, for all variables x occurring in C, the only Boolean green subterms of $x\theta$ are $\top$ and $\bot$. Since $u\theta$ is a fully applied logical symbol different from $\top$ and $\bot$, the term u cannot be a variable. Eligibility of p in $C\theta$ implies eligibility of p' in C by Lemma 65. If u is a fluid term, by conditions A and B, it must be in a literal $u \approx \top$ or $u \approx \bot$ or $u \approx v$ of C, for some variable-headed term v.

- BOOLRW: Let (t, t') be the pair used among the ones listed for BOOLRW. Then we can extend θ to the variables in t such that the resulting substitution θ' is a unifier of t and u. Therefore, there exists an idempotent $\sigma \in \mathrm{CSU}(t, u)$ such that for some substitution ρ and for all variables x in C, we have $x\sigma\rho = x\theta'$. Thus, there is the following BOOLRW inference $\iota' \in \mathit{HInf}$:

$$\frac{C\langle u\rangle}{C\langle t'\rangle\sigma}\ \textsc{BoolRw}$$

In this case, ι is the $\sigma\rho$-ground instance of ι' and is therefore liftable.

- GFORALLRW: Then $u\theta = \forall\langle\tau\rangle v$ and the inference ι is of the form

  $$\frac{C\theta\langle\forall\langle\tau\rangle v\rangle_p}{C\theta\langle v\,\mathit{GHWit}(C\theta,p)\rangle_p}\ \textsc{GForallRw}$$

  for some term v and some type τ.
  Let β be a type variable and y a variable of type $\beta \to \mathrm{o}$. We define a substitution θ' mapping y to v, β to τ, and all other variables x to $x\theta$. Then $(\forall\langle\beta\rangle\, y)\theta' = \forall\langle\tau\rangle\, v = u\theta = u\theta'$ and hence θ' is a unifier of $\forall\langle\beta\rangle\, y$ and u. Hence, there exists an idempotent $\sigma \in \mathrm{CSU}(\forall\langle\beta\rangle\, y,\, u)$ such that for some substitution ρ, for all variables x in C, and for $x \in \{\beta, y\}$, we have $x\sigma\rho = x\theta'$. If $\mathcal{F}(C\theta\langle\top\rangle_p) = \mathcal{F}(C\langle\top\rangle_{p'}\theta)$ is not a tautology, the affected literal in C cannot be of the form $u \approx \top$. Thus there exists the following inference $\iota' \in \mathit{HInf}$:

  $$\frac{C\langle u\rangle}{C\langle y\,(\mathsf{sk}_{\Pi\bar\alpha.\,\forall\bar x.\,\exists z.\,\neg(y\sigma\, z)}\langle\bar\alpha\rangle\,\bar x)\rangle\sigma}\ \textsc{ForallRw}$$

  We have $\mathit{GHWit}(C\theta,p) = \mathsf{sk}_{\Pi\bar\alpha.\,\forall\bar x.\,\exists z.\,\neg(y\sigma\, z)}\langle\bar\alpha\rangle\,\bar x\theta$ by definition of the witness function, where $\bar\alpha$ are the free type variables and $\bar x$ are the free variables occurring in $y\sigma$ in order of first occurrence. Hence, ι is the $\sigma\rho$-ground instance of ι' and is therefore liftable.
- GEXISTSRW: Analogous to GFORALLRW.
- EQHOIST: Let x and y be fresh variables. Then we can extend θ to a x and y such that the resulting substitution θ' is a unifier of u and $x \mathrel{\approx\!\!\!\approx} y$. Thus, there exists an idempotent $\sigma \in \mathrm{CSU}(u,\, x \mathrel{\approx\!\!\!\approx} y)$ such that for some substitution ρ, for all variables z in C, and for $z \in \{x, y\}$, we have $z\sigma\rho = z\theta'$. Hence, there is the following EQHOIST inference $\iota' \in \mathit{HInf}$:

  $$\frac{C\langle u\rangle}{(C\langle\bot\rangle \vee x \approx y)\sigma}\ \textsc{EqHoist}$$

  Then ι is the $\sigma\rho$-ground instance of ι' and is therefore liftable.
- NEQHOIST: Analogous to EQHOIST.
- GFORALLHOIST: Let α be a fresh type variable and y be a fresh variable. Then we can extend θ to y such that the resulting substitution θ' is a unifier of u and $\forall\langle\alpha\rangle\, y$. Thus, there exists an idempotent $\sigma \in \mathrm{CSU}(u,\, \forall\langle\alpha\rangle\, y)$ such that for some substitution ρ, for all variables x in C, and for $x = y$, we have $x\sigma\rho = x\theta'$. Thus, there is the following FORALLHOIST inference $\iota' \in \mathit{HInf}$:

  $$\frac{C\langle u\rangle}{(C\langle\bot\rangle \vee y\,x \approx \top)\sigma}\ \textsc{ForallHoist}$$

  Then ι is the $\sigma\rho$-ground instance of ι' and is therefore liftable.

- GExistsHoist: Analogous to GForallHoist.

Redundant Case: Neither condition A nor B holds. Then p corresponds to a position in C at or below a fluid term, but p is not the position of v in a literal $v \approx \top$ or $v \approx \bot$. Let $p = p_1.p_2$ such that p_1 is the longest prefix of p that corresponds to a green position p_1' in C. Let $u = C|_{p_1'}$. Let z and x be fresh variables. Define a substitution θ' that maps the variable z to $\lambda y.(u\theta)\langle y\rangle_{p_2}$, the variable x to $u\theta|_{p_2}$, and all other variables w to $w\theta$. Then $(z\,x)\theta' = (u\theta)\langle u\theta|_{p_2}\rangle_{p_2} = u\theta = u\theta'$. So θ' is a unifier of $z\,x$ and u. Thus, there exists an idempotent $\sigma \in \mathrm{CSU}(z\,x,\, u)$ such that for some substitution ρ, for all variables y in C, and for $y \in \{z, x\}$, we have $y\sigma\rho = y\theta'$. For all of the inference rules, $C\theta|_p = u\theta|_{p_2}$ cannot be $\bot$ or $\top$. Thus, $x\theta' \neq \bot, \top$ and therefore $x\sigma \neq \bot, \top$. Hence, we have $(z\,\bot)\theta' = (u\theta)\langle\bot\rangle_{p_2} \neq (u\theta)\langle x\theta'\rangle_{p_2} = (z\,x)\theta'$ and therefore $(z\,\bot)\sigma \neq (z\,x)\sigma$. Analogously, we have $(z\,\top)\sigma \neq (z\,x)\sigma$. The position p_1 must be eligible in $C\theta$ because p is eligible in $C\theta$ and p_1 is the longest prefix of p that corresponds to a green position p_1' in C. Eligibility of p_1 in $C\theta$ implies eligibility of p_1' in C by Lemma 65. Then there are the following inferences ι_{bool} and ι_{loob} from C:

$$\frac{C\langle u\rangle_{p_1'}}{(C\langle z\,\bot\rangle_{p_1'} \vee x \approx \top)\sigma}\;\textsc{FluidBoolHoist}$$

$$\frac{C\langle u\rangle_{p_1'}}{(C\langle z\,\top\rangle_{p_1'} \vee x \approx \bot)\sigma}\;\textsc{FluidLoobHoist}$$

Since N is saturated w.r.t. *HInf* and $HRed_{\mathrm{I}}$, these inferences are in $HRed_{\mathrm{I}}(N)$. We have

$$\begin{aligned} &\mathcal{F}((C\langle z\,\bot\rangle_{p_1'} \vee x \approx \top)\theta') = \mathcal{F}(C\theta\langle\bot\rangle_p \vee C\theta|_p \approx \top) \\ \text{and}\quad &\mathcal{F}((C\langle z\,\top\rangle_{p_1'} \vee x \approx \bot)\theta') = \mathcal{F}(C\theta\langle\top\rangle_p \vee C\theta|_p \approx \bot) \end{aligned}$$

These two clauses entail $\mathcal{F}(C\theta)$. Since p is not the position of v in a literal $v \approx \top$ or $v \approx \bot$, the two clauses are also smaller than $\mathcal{F}(C\theta)$. Since $\iota_{\mathrm{bool}} \in HRed_{\mathrm{I}}(N)$, we have $\mathcal{G}^q(\iota_{\mathrm{bool}}) \subseteq GHRed_{\mathrm{I}}(\mathcal{G}(N))$ and therefore the clauses $\mathcal{F}(\mathcal{G}(N))$ that are smaller than $C\theta' = C\theta$ entail $\mathcal{F}((C\langle z\,\bot\rangle_{p_1'} \vee x \approx \top)\theta')$. Similarly, since $\iota_{\mathrm{loob}} \in HRed_{\mathrm{I}}(N)$, we have $\mathcal{G}(concl(\iota_{\mathrm{loob}})) \subseteq \mathcal{G}(N) \cup GHRed_{\mathrm{C}}(\mathcal{G}(N))$. Therefore $\mathcal{F}((C\langle z\,\top\rangle_{p_1'} \vee x \approx \bot)\theta')$ is entailed by clauses in $\mathcal{G}(N)$ that are smaller than or equal to itself. Thus, $C\theta$ is redundant and therefore ι is redundant. Here, it is crucial that we consider inferences with a redundant premise as redundant. ☐

By the above lemmas, every *HInf* inference is liftable or redundant. Using these lemmas, we can apply Theorem 64 to lift ground refutational completeness to non-ground refutational completeness.

Lemma 69 (Static refutational completeness w.r.t. $\models_{\mathcal{G}}$). *The inference system HInf is statically refutationally complete w.r.t. $\models_{\mathcal{G}}$ and $(HRed_{\mathrm{I}}, HRed_{\mathrm{C}})$. In other words, if $N \subseteq \mathcal{C}_{\mathrm{H}}$ is a clause set saturated w.r.t. HInf and $HRed_{\mathrm{I}}$, then $N \models_{\mathcal{G}} \bot$ if and only if $\bot \in N$.*

Proof. We want to apply Theorem 64. GHInf^q is statically refutationally complete for all $q \in Q$ by Theorem 60. By Lemmas 66, 67, and 68, for every saturated $N \subseteq \mathcal{C}_\mathrm{H}$, there exists $q \in \mathcal{G}(Q)$ such that all inferences $\iota \in \mathit{GHInf}^q$ with $\mathit{prems}(\iota) \in \mathcal{G}(N)$ either are $\mathcal{G}^q$-ground instances of HInf-inferences from N or belong to $\mathit{GHRed}^q_\mathrm{I}(\mathcal{G}(N))$. Thus, Theorem 64 applies. □

Dynamic refutational completeness is easy to derive from static refutational completeness.

Lemma 70 (Dynamic refutational completeness w.r.t. $\models_\mathcal{G}$). *The inference system HInf is dynamically refutationally complete w.r.t. $\models_\mathcal{G}$ and $(\mathit{HRed}_\mathrm{I}, \mathit{HRedC})$, as per Definition 48.*

Proof. By Theorem 17 of the saturation framework, this follows from Lemma 69. □

To derive a corresponding result for the entailment relation $\models$, we employ the following lemma, which states equivalence of Herbrand entailment $\models_\mathcal{G}$ and Tarski entailment $\models$ on $\mathsf{Q}_\approx$-normal clauses.

Lemma 71. *Let $N \subseteq \mathcal{C}_\mathrm{H}$ be $\mathsf{Q}_\approx$-normal. Then we have $N \models_\mathcal{G} \bot$ if and only if $N \models \bot$.*

Proof. By Lemma 30, any model of N is also a model of $\mathcal{G}(N)$. So $N \models_\mathcal{G} \bot$ implies $N \models \bot$.

For the other direction, let $\mathfrak{I}$ be a model of $\mathcal{G}(N)$. We must show that there exists a model of N. Let $\mathfrak{I}'$ be the interpretation obtained from $\mathfrak{I}$ by removing all domains that cannot be expressed as $[\![\tau]\!]_{\mathfrak{I}_\mathsf{ty}}$ for some ground type τ and by removing all domain elements that cannot be expressed as $[\![t]\!]_\mathfrak{I}$ for some ground term t. We restrict the type interpretation function $\mathcal{J}_\mathsf{ty}$, the interpretation function $\mathcal{J}$, and the λ-designation function $\mathcal{L}$ of $\mathfrak{I}$ accordingly.

The restriction $\mathcal{J}'$ of $\mathcal{J}$ still maps the logical symbols correctly: For most logical symbols, this is obvious. Only $\forall$ and $\exists$ deserve some further explanations. For all domains $\mathcal{D}$ of $\mathfrak{I}$, we have $\mathcal{J}(\exists, \mathcal{D})(f) = \max\{f(a) \mid a \in \mathcal{D}\}$. For the corresponding domain $\mathcal{D}' \subseteq \mathcal{D}$, if it has not been removed entirely, we have just defined $\mathcal{J}'(\exists, \mathcal{D}')(f) = \mathcal{J}(\exists, \mathcal{D})(f)$. We must show that $\mathcal{J}'(\exists, \mathcal{D}')(f) = \max\{f(a) \mid a \in \mathcal{D}'\}$ for all f that can be expressed as $[\![t]\!]_\mathfrak{I}$ for some ground term t. This claim can only be violated if there exist $a \in \mathcal{D}$ with $f(a) = 1$ and if all of them have been removed in $\mathcal{D}'$. But we have not removed all such elements a because one of them can be expressed as $[\![\varepsilon\, t]\!]_\mathfrak{I}$ where t is the ground term such that $f = [\![t]\!]_\mathfrak{I}$. We can argue similarly for $\forall$.

Clearly, all terms have the same denotation in $\mathfrak{I}'$ as in $\mathfrak{I}$. Thus, the truth values of ground clauses are identical in $\mathfrak{I}$ and $\mathfrak{I}'$. Since $\mathfrak{I}$ is proper, $\mathfrak{I}'$ is also proper. Hence, $\mathfrak{I} \models \mathcal{G}(N)$ implies $\mathfrak{I}' \models \mathcal{G}(N)$.

It remains to show that $\mathfrak{I}' \models N$. Let $C \in N$ and let ξ be a valuation for $\mathfrak{I}'$. We must show that C is true in $\mathfrak{I}'$ under ξ. By assumption, C is $\mathsf{Q}_\approx$-normal.

By the construction of $\mathfrak{I}'$, there is a grounding substitution θ such that for all type variables α and all term variables x occurring in C, we have $\xi(\alpha) = [\![\alpha\theta]\!]_{\mathcal{J}'_\mathsf{ty}}$ and $\xi(x) = [\![x\theta]\!]_{\mathfrak{I}'}$. Then, by Lemma 29, $[\![t\theta]\!]_{\mathfrak{I}'} = [\![t]\!]^\xi_{\mathfrak{I}'}$ for all subterms t of C. Moreover, we can choose θ such that for all variables x, the only Boolean green subterms of $x\theta$ are $\top$

and $\bot$ and such that $x\theta$ is $\mathsf{Q}_{\approx}$-normal. If $x\theta$ contains a Boolean green subterm different from $\top$ and $\bot$, we can replace it by $\top$ or $\bot$ while preserving its denotation and thus the property that $[\![t\theta]\!]_{\mathcal{J}'} = [\![t]\!]^{\xi}_{\mathcal{J}'}$ for all subterms t of C. If $x\theta$ is not $\mathsf{Q}_{\approx}$-normal, we $\mathsf{Q}_{\approx}$-normalize it, which preserves the denotation of terms by Lemma 53.

By Lemma 9, it follows that $C\theta$ is $\mathsf{Q}_{\approx}$-normal. Thus, $C\theta \in \mathcal{G}(C)$. Since $\mathcal{I}' \models \mathcal{G}(C)$ and thus $C\theta$ is true in $\mathcal{I}'$, also C is true in $\mathcal{I}'$ under ξ because $[\![t\theta]\!]_{\mathcal{J}'} = [\![t]\!]^{\xi}_{\mathcal{J}'}$ for all subterms t of C. □

Using this lemma, we can derive the following theorem, which is essentially dynamic refutational completeness with the caveat that the initial clause set must be $\mathsf{Q}_{\approx}$-normal, which in practice can be fulfilled by $\mathsf{Q}_{\approx}$-normalizing the input problem in preprocessing.

Theorem 72 (Dynamic refutational completeness w.r.t. $\models$). *Let $(N_i)_i$ be a derivation w.r.t. $\mathit{HRed}_{\mathrm{C}}$, as per Definition 48, such that N_0 is $\mathsf{Q}_{\approx}$-normal and $N_0 \models \bot$. Moreover, assume that $(N_i)_i$ is fair w.r.t. HInf and $\mathit{HRed}_{\mathrm{I}}$. Then we have $\bot \in N_i$ for some i.*

Proof. This is a consequence of Lemmas 70 and 71. □

To derive a similar result for $\models_{\approx}$, we need the following lemma:

Lemma 73. *Let $N_0 \subseteq \mathcal{C}_{\mathrm{H}}$ be a clause set that does not contain any* sk *symbols. If $N_0 \models_{\approx} \bot$, then $N_0 \models \bot$.*

Proof. Equivalently, the lemma statement can be formulated as follows: If N_0 does not have Skolem-aware models, it does not have models at all. We assume that N_0 has a model $\mathcal{I}$ and must show that there exists a Skolem-aware model $\mathcal{I}'$ of N_0.

To transform the model $\mathcal{I} = (\mathcal{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$ into a skolem-aware model $\mathcal{I}' = (\mathcal{I}'_{\mathsf{ty}}, \mathcal{J}', \mathcal{L}')$, we redefine the interpretation of the Skolem symbol $\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,t\,z} : \Pi\bar{\alpha}.\,\bar{\tau} \to \upsilon$ as follows. Given some domains $\bar{\mathcal{D}}$, let $\xi(\bar{\alpha}) = \bar{\mathcal{D}}$. Then define $\mathcal{J}'(\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,t\,z}, \bar{\mathcal{D}}) = [\![\lambda\bar{x}.\,\varepsilon\langle\upsilon\rangle\,t]\!]^{\xi}_{\mathcal{J}}$ and $\mathcal{L}'(\xi, \lambda x.s) = \mathcal{L}(\xi, \lambda x.s')$ where s' is obtained from s by replacing each occurrence of a subterm $\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,t\,z}\langle\bar{\upsilon}\rangle$ by $(\lambda\bar{x}.\,\varepsilon\langle\upsilon\rangle\,t)\{\bar{\alpha} \mapsto \bar{\upsilon}\}$. This modification of $\mathcal{I}$ yields a new interpretation $\mathcal{I}'$, which is still a model of N_0 because N_0 does not contain any sk symbols. Moreover, it is a Skolem-aware model of N_0 because our redefinition ensures $\mathcal{I}' \models (\exists\langle\upsilon\rangle\,(\lambda z.\,t\,z)) \approx t\,(\mathsf{sk}_{\Pi\bar{\alpha}.\,\forall\bar{x}.\,\exists z.\,t\,z}\langle\bar{\alpha}\rangle\,\bar{x})$. □

Using this lemma, we can derive dynamic refutational completeness for $\models_{\approx}$ with additional assumptions on $\mathsf{Q}_{\approx}$-normality and the absence of sk-symbols. These assumptions can be fulfilled by $\mathsf{Q}_{\approx}$-preprocessing and by making the sk symbols internal such that they can not be expressed by the input language of the prover.

Theorem 74 (Dynamic refutational completeness w.r.t. $\models_{\approx}$). *Let $(N_i)_i$ be a derivation w.r.t. $\mathit{HRed}_{\mathrm{C}}$, as defined in Definition 48, such that N_0 is $\mathsf{Q}_{\approx}$-normal, N_0 does not contain any* sk *symbols, and $N_0 \models_{\approx} \bot$. Moreover, assume that $(N_i)_i$ is fair w.r.t. HInf and $\mathit{HRed}_{\mathrm{I}}$. Then we have $\bot \in N_i$ for some i.*

Proof. By Lemma 73, $N_0 \models \bot$. Hence, Theorem 72 applies. □

5 Implementation

We implemented our calculus in the Zipperposition prover[4], whose OCaml source code makes it convenient to prototype calculus extensions. It has also assimilated some of E's [29] heuristics, which helps it achieve strong performance at the CASC competition.

Like the calculus, its implementation extends the implementations of λSup and oSup. From the former, we inherit the given clause loop which supports enumerating infinitely many inference conclusions, calculus extensions, and higher-order heuristics [32]. From the latter, we inherit the encoding of negative predicate literals as $s \approx \bot$ and a basis for the implementation of FALSEELIM, BOOLRW, FORALLRW, EXISTSRW, and all HOIST rules.

The implementation of oSup relies heavily on BOOLSIMP, and we keep this rule as the basis of our Boolean simplification machinery. As a result, BOOLRW can be reduced to two cases: (1) all arguments $\bar{s}_n$ of $u = h\,\bar{s}_n$ are variable-headed terms and thus we must unify s_i with all combinations of $\top$ and $\bot$; or (2) either $u = s \mathbin{\boldsymbol{\approx}} t$ or $u = s \mathbin{\boldsymbol{\not\approx}} t$ and thus we must compute the unifiers of s and t.

As in the implementation of λSup, we approximate fluid terms as terms that are either nonground λ-expressions or terms of the form $x\,\bar{s}_n$ with $n > 0$. Two slight, accidental discrepancies are that we also count variable occurrences below quantifiers as deep and perform EFACT inferences even if the maximal literal is selected. Since we expect FLUIDBOOLHOIST and FLUIDLOOBHOIST to be highly explosive, we penalize them and all of their offspring. In addition to various λSup extensions [7, Sect. 5], we also use all the rules for Boolean reasoning described by Vukmirović and Nummelin [34] except for the BOOLEF rules. The BoolEF rules have been removed from Zipperposition because they did not seem to improve the prover's performance.

Our implementation is a graceful extension of oSup. For the rules EQHOIST, NEQHOIST, FORALLHOIST, and EXISTSHOIST, if the subterm u is not variable-headed, there is an obvious most general unifier, which allows us to avoid invoking the unification procedure and to generate the conclusion directly, as in oSup.

6 Evaluation

We evaluate our implementation and compare it with other higher-order provers. Our experiments were performed on StarExec Miami servers equipped with Intel Xeon E5-2620 v4 CPUs clocked at 2.10 GHz. We used all 2606 TH0 theorems from the TPTP 7.3.0 library [31] and 1253 "Judgment Day" problems [12] generated using Sledgehammer (SH) [28] as our benchmark set. An archive containing the benchmarks and the raw evaluation results is publicly available.[5] We divide the evaluation in two parts: evaluation of the calculus rules and comparison with other higher-order provers.

Calculus Evaluation. In this first part, we evaluate selected parameters of Zipperposition by varying only the studied parameter in a fixed well-performing configuration. This base configuration disables axioms (CHOICE) and (EXT) and the FLUID- rules. It

[4] https://github.com/sneeuwballen/zipperposition
[5] https://doi.org/10.5281/zenodo.4534759

uses the unification procedure of Vukmirović et al. [33] in its complete variant—i.e., the variant that produces a complete set of unifiers. It uses none of the early Boolean rules described by Vukmirović and Nummelin [34]. The preprocessor $\mathsf{Q}_{\approx}$ is disabled as well. All of the completeness-preserving simplification rules described in Sect. 3.7 are enabled, except for the simplifying BOOLHOIST (combined with LOOBHOIST). The configuration uses immediate clausification. We set the CPU time limit to 30 s in all three experiments.

In the first experiment, we assess the overhead incurred by the FLUID- rules. These rules unify with a term whose head is a fresh variable. Thus, we expected that they needed to be tightly controlled to achieve good performance. To test our hypothesis, we simultaneously modified the parameters of these three rules. In Figure 1, the *off* mode simply disables the rules, the *pragmatic* mode uses a terminating incomplete unification algorithm (the pragmatic variant of Vukmirović et al. [33]), and the *complete* mode uses a complete unification algorithm. The results show that disabling FLUID- rules altogether achieves the best performance. When the complete variant of the unification algorithm is used, inferences are scheduled in a queue designed to postpone explosive inferences [7, Sect. 6]. In contrast, in the pragmatic variant, a terminating algorithm is employed, but still flooding the proof state with FLUID- rules conclusions severely hinders performance. Even though enabling FLUID- rules degrades performance overall, *complete* finds 35 proofs not found by *off*, and *pragmatic* finds 22 proofs not found by *off*. On Sledgehammer benchmarks, this effect is much weaker, likely because the Sledgehammer benchmarks require less higher-order reasoning: *complete* finds only one new proof over *off*, and *pragmatic* finds only four.

In the second experiment, we explore the clausification methods introduced at the end of Sect. 3: *inner* delayed clausification, which relies on the core calculus to reason about logical symbols; *outer* delayed clausification, which clausifies step-by-step guided by the outermost logical symbols; and *immediate* clausification, which eagerly applies a monolithic clausification algorithm when encountering top-level logical symbols. The modes *inner* and *outer* employ oSup's RENAME rule, which renames Boolean terms headed by logical symbols using a Tseitin-like transformation if they occur at least four times in the proof state. Vukmirović and Nummelin [34] observed that *outer* clausification can greatly help prove higher-order problems, and we expected it to perform well for our calculus, too. The results are shown in Figure 2. The results confirm our hypothesis: The *outer* mode outperforms *immediate* on both TPTP and Sledgehammer benchmarks. The *inner* mode performs worst, but on Sledgehammer benchmarks, it proves 17 problems beyond the reach of the other two. Looking at the proofs found by *inner*, we observed a pattern: in many cases (e.g., for the benchmarks `prob_295__3252866_1`, `prob_296__3252872_1`, `prob_366__5338318_1`, `prob_419__5371618_1`) the problems contain axioms of the form $\phi \rightarrow \psi$. When such axioms are not clausified, superposition and demodulation can often reduce either ϕ or ψ to $\top$ or $\bot$. At this point, simplification rules will act on the resulting term, simplifying it enough that the proof can easily be found.

In the third experiment, we investigate the effect of axiom (CHOICE), which is necessary to achieve refutational completeness. To evaluate (CHOICE), we either disabled it in a configuration labeled *off* or set the axiom's penalty p to different values. In Zipper-

	off	*pragmatic*	*complete*
TPTP	**1642**	1591	1619
SH	**467**	431	437

Fig. 1. Evaluation of FLUID- rules

	inner	*outer*	*immediate*
TPTP	1323	**1670**	1642
SH	406	**470**	467

Fig. 2. Evaluation of clausification method

	off	$p = 64$	$p = 16$	$p = 4$	$p = 1$
TPTP	**1642**	1617	1613	1615	1594
SH	**467**	458	458	459	445

Fig. 3. Evaluation of axiom (CHOICE)

	TPTP	ofSH	SH
CVC4	1796	680	619
Leo-III	2104	681	621
Satallax	2162	573	587
Vampire	2131	692	681
Zip	2301	**734**	**736**
New Zip	**2320**	724	720
Leo-III-uncoop	1619	223	240
Satallax-uncoop	2038	467	482
Zip-uncoop	2223	667	673
New Zip-uncoop	2236	640	644

Fig. 4. Evaluation of all competitive higher-order provers

position, penalties are propagated through inference and simplification rules and are used to increase the heuristic weight of clauses, postponing the selection of penalized clauses. The results are shown in Figure 3. As expected, disabling (CHOICE), or at least penalizing it heavily, improves performance. Yet enabling (CHOICE) can be crucial: For 19 TPTP problems, the proofs are found when (CHOICE) is enabled and $p = 4$, but not when the rule is disabled. On Sledgehammer problems, this effect is weaker, with only two new problems proved for $p = 4$.

Prover Comparison. In this second part, we compare Zipperposition's performance with other higher-order provers. Like at CASC-J10, the wall-clock time limit was 120 s, the CPU time limit was 960 s, and the provers were run on StarExec Miami. We used the following versions of all systems that took part in the THF division: CVC4 1.8 [5], Leo-III 1.5.2 [30], Satallax 3.5 [13], and Vampire 4.5 [10]. The developers of Vampire have informed us that its higher-order schedule is optimized for running on a single core. As a result, the prover suffers some degradation of performance when running on multiple cores. We evaluate both the version of Zipperposition that took part in CASC-J10 (*Zip*) and the updated version of Zipperposition that supports our new calculus (*New Zip*). Zip's portfolio of prover configurations is based on λSup and techniques described by Vukmirović and Nummelin [34]. New Zip's portfolio is specially designed for our new calculus and optimized for TPTP problems. The portfolio does not contain a complete configuration, but the combination of configurations is close to complete. Leo-III, Satallax, and Zipperposition are cooperative theorem provers: They invoke a backend reasoner to finish the proof attempt. To test the performance of their calculi in isolation, we also invoked them in uncooperative mode. To assess the performance of Boolean reasoning, we used Sledgehammer benchmarks generated both with native Booleans (SH) and with an encoding into Boolean-free higher-order logic (ofSH). For technical reasons, the encoding also performs λ-lifting, but this minor transformation should have little impact on results [7, Sect. 7].

The results are shown in Figure 4. The updated version of New Zip beats Zip on TPTP problems but lags behind Zip on Sledgehammer benchmarks as we have yet to further explore more general heuristics that work well with our new calculus. The Sledgehammer benchmarks fail to demonstrate the superiority of native Booleans reasoning compared with an encoding, and in fact CVC4 and Leo-III perform dramatically better on the encoded Boolean problems, suggesting that there is room for tuning.

The uncooperative versions of Zipperposition show strong performance on both benchmark sets. This suggests that, with thorough parameter tuning, higher-order superposition outperforms tableaux, which had been the state of the art in higher-order reasoning for a decade. Without backend reasoners, Zipperposition proves fewer Sledgehammer problems than Vampire. We conjecture that implementing our calculus in Vampire or E would remove the need for a backend reasoner and make the calculus even more useful in practice.

7 Conclusion

We have created a superposition calculus for higher-order logic and proved it sound and refutationally complete. Most of the key ideas have been developed in previous work by us and colleagues, but combining them in the right way has been challenging. A key idea was to $\mathsf{Q}_{\approx}$-normalize away inconvenient terms.

Unlike earlier refutationally complete calculi for full higher-order logic based on resolution or paramodulation, our calculus employs a term order, which restricts the proof search, and a redundancy criterion, which can be used to add various simplification rules while keeping refutational completeness. These two mechanisms are undoubtedly major factors in the success of first-order superposition, and it is very fortunate that we could incorporate both in a higher-order calculus. An alternative calculus with the same two mechanisms could be achieved by combining oSup with Bhayat and Reger's combinatory superposition [9]. The article on λSup [7, Sect. 8] discusses related work in more detail.

The evaluation results show that our calculus is an excellent basis for higher-order theorem proving. In future work, we want to experiment further with the different parameters of the calculus (for example, with Boolean subterm selection heuristics) and implement it in a state-of-the-art prover such as E.

Acknowledgment. Uwe Waldmann provided advice and carefully checked the completeness proof. Visa Nummelin led the design of the oSup calculus, which was a crucial stepping stone for this work. Simon Cruanes helped us with the implementation. Martin Desharnais generated (and regenerated) the Sledgehammer benchmarks. Ahmed Bhayat explained the higher-order Vampire schedule and suggested textual improvements. Christoph Benzmüller provided us insight into the pitfalls of higher-order reasoning. Alexander Steen explained the details of Leo-III scheduling. Geoff Sutcliffe let us use StarExec Miami for our experiments. Simon Cruanes, Herman Geuvers, Mathias Fleury, Nicolas Peltier, Giles Reger, Mark Summerfield, and the anonymous reviewers suggested textual improvements. We thank them all.

Bentkamp, Blanchette, and Vukmirović'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). Blanchette's research has received funding from the Netherlands Organization for Scientific Research (NWO) under the Vidi program (project No. 016.Vidi.189.037, Lean Forward).

References

[1] Andrews, P.B.: On connections and higher-order logic. J. Autom. Reason. **5**(3), 257–291 (1989)
[2] Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. **4**(3), 217–247 (1994)
[3] Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 19–99. Elsevier and MIT Press (2001)
[4] Bachmair, L., Ganzinger, H., Lynch, C., Snyder, W.: Basic paramodulation and superposition. In: Kapur, D. (ed.) CADE-11. LNCS, vol. 607, pp. 462–476. Springer (1992)
[5] Barrett, C.W., Conway, C.L., Deters, M., Hadarean, L., Jovanovic, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: CAV. LNCS, vol. 6806, pp. 171–177. Springer (2011)
[6] Bentkamp, A., Blanchette, J., Cruanes, S., Waldmann, U.: Superposition for lambda-free higher-order logic. Log. Meth. Comput. Sci. **17**(2), 1:1–1:38 (2021)
[7] Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P., Waldmann, U.: Superposition with lambdas, accepted in J. Autom. Reason. Preprint at https://arxiv.org/abs/2102.00453v1 (2021)
[8] 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)
[9] Bhayat, A., Reger, G.: Set of support for higher-order reasoning. In: Konev, B., Urban, J., Rümmer, P. (eds.) PAAR-2018. CEUR Workshop Proceedings, vol. 2162, pp. 2–16. CEUR-WS.org (2018)
[10] Bhayat, A., Reger, G.: A combinator-based superposition calculus for higher-order logic. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020, Part I. LNCS, vol. 12166, pp. 278–296. Springer (2020)
[11] Blanqui, F., Jouannaud, J.P., Rubio, A.: The computability path ordering. Log. Meth. Comput. Sci. **11**(4) (2015)
[12] Böhme, S., Nipkow, T.: Sledgehammer: Judgement Day. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS, vol. 6173, pp. 107–121. Springer (2010)
[13] 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)
[14] Cervesato, I., Pfenning, F.: A linear spine calculus. J. Log. Comput. **13**(5), 639–688 (2003)
[15] Fitting, M.: Types, Tableaus, and Gödel's God. Kluwer (2002)
[16] Ganzinger, H., Stuber, J.: Superposition with equivalence reasoning and delayed clause normal form transformation. Information and Computation **199**(1–2), 3–23 (2005)
[17] Gordon, M.J.C., Melham, T.F. (eds.): Introduction to HOL: A Theorem Proving Environment for Higher Order Logic. Cambridge University Press (1993)

[18] Huet, G.P.: A mechanization of type theory. In: Nilsson, N.J. (ed.) IJCAI-73. pp. 139–146. William Kaufmann (1973)

[19] Huet, G.P.: A unification algorithm for typed lambda-calculus. Theor. Comput. Sci. **1**(1), 27–57 (1975)

[20] Jensen, D.C., Pietrzykowski, T.: Mechanizing ω-order type theory through unification. Theor. Comput. Sci. **3**(2), 123–171 (1976)

[21] Jouannaud, J.P., Rubio, A.: Rewrite orderings for higher-order terms in eta-long beta-normal form and recursive path ordering. Theor. Comput. Sci. **208**(1–2), 33–58 (1998)

[22] Kaliszyk, C., Sutcliffe, G., Rabe, F.: TH1: The TPTP typed higher-order form with rank-1 polymorphism. In: Fontaine, P., Schulz, S., Urban, J. (eds.) PAAR-2016. CEUR Workshop Proceedings, vol. 1635, pp. 41–55. CEUR-WS.org (2016)

[23] Kőnig, D.: Über eine Schlussweise aus dem Endlichen ins Unendliche. Acta Sci. Math. (Szeged) **3499/2009**(3:2–3), 121–130 (1927)

[24] Kotelnikov, E., Kovács, L., Suda, M., Voronkov, A.: A clausal normal form translation for FOOL. In: Benzmüller, C., Sutcliffe, G., Rojas, R. (eds.) GCAI 2016. EPiC, vol. 41, pp. 53–71. EasyChair (2016)

[25] Ludwig, M., Waldmann, U.: An extension of the Knuth-Bendix ordering with LPO-like properties. In: Dershowitz, N., Voronkov, A. (eds.) LPAR-14. LNCS, vol. 4790, pp. 348–362. Springer (2007)

[26] Mayr, R., Nipkow, T.: Higher-order rewrite systems and their confluence. Theor. Comput. Sci. **192**(1), 3–29 (1998)

[27] Nummelin, V., Bentkamp, A., Tourret, S., Vukmirović, P.: Superposition with first-class Booleans and inprocessing clausification (technical report), https://matryoshka-project.github.io/pubs/boolsup_report.pdf

[28] Paulson, L.C., Blanchette, J.C.: Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In: Sutcliffe, G., Schulz, S., Ternovska, E. (eds.) IWIL-2010. EPiC, vol. 2, pp. 1–11. EasyChair (2012)

[29] Schulz, S.: E - a brainiac theorem prover. AI Commun. **15**(2-3), 111–126 (2002)

[30] 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)

[31] Sutcliffe, G.: The TPTP problem library and associated infrastructure—from CNF to TH0, TPTP v6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017)

[32] Vukmirović, P., Bentkamp, A., Blanchette, J., Cruanes, S., Nummelin, V., Tourret, S.: Making higher-order superposition work. In: Platzer, A., Sutcliffe, G. (eds.) CADE-28. LNCS, Springer (2021)

[33] Vukmirović, P., Bentkamp, A., Nummelin, V.: Efficient full higher-order unification. In: Ariola, Z.M. (ed.) FSCD 2020. LIPIcs, vol. 167, pp. 5:1–5:17. Schloss Dagstuhl—Leibniz-Zentrum für Informatik (2020)

[34] Vukmirović, P., Nummelin, V.: Boolean reasoning in a higher-order superposition prover. In: PAAR-2020. CEUR Workshop Proceedings, vol. 2752, pp. 148–166. CEUR-WS.org (2020)

[35] Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020, Part I. LNCS, vol. 12166, pp. 316–334. Springer (2020)