SIMPLIFIED VARIANT OF OPTIMISTIC LAMBDA-SUPERPOSITION

This document describes a simplified variant of the optimistic λ-superposition calculus. The main difference is that the present variant does not annotate clauses with constraints. This simplifies especially the completness proof because we can use ground clauses instead of ground closures in the first-order part of the proof. It also strengthens and simplifies the redundancy criterion. However, we are forced to introduce superposition inferences into variables when those variables also have occurrences inside parameters.

1. Logic

Our formalism is higher-order logic with functional and Boolean extensionality, rank-1 polymorphism, but without choice and the axiom of infinity. The logic closely resembles Gordon and Melham's HOL [14], the TPTP TH1 standard [15], and the logic underlying λ-superposition by Bentkamp et al. [5].

Departing from Bentkamp et al., in the present work, quantifiers are not supported and must always be encoded as $(\lambda x.\, t) \approx (\lambda x.\, \mathsf{T})$ and $(\lambda x.\, t) \not\approx (\lambda x.\, \bot)$. This is necessary because quantifiers would prevent us from constructing a suitable term order for the extensionality behavior that we want to achieve. Moreover, we do not include the axiom of choice.

To make the positive literal of the extensionality axiom maximal, we introduce a special type of argument to constants into our syntax, the *parameters*. A constant that takes parameters cannot occur without them; partial application is not allowed for parameters. Moreover, parameters cannot contain variables bound by λ-abstractions.

As our semantics, we use Henkin semantics. True statements in these semantics correspond exactly to provable statements in the HOL systems. Since Henkin semantics is not subject to Gödel's first incompleteness theorem, it allows us to prove refutational completeness.

1.1. Syntax.

We use the notation $\bar{a}_n$ or $\bar{a}$ to denote a tuple $(a_1, \ldots, a_n)$. If f is a unary function, we write $f(\bar{a}_n)$ for the elementwise application $(f(a_1), \ldots, f(a_n))$.

1.1.1. *Types.*

To define our logic's types, we fix an infinite set $\mathcal{V}_{\mathsf{ty}}$ of type variables. A set Σ_{ty} of type constructors, each associated with an arity, is a *type signature* if it contains at least one nullary type constructor o of Booleans and a binary type constructor $\to$ of functions. A *type* is either a type variable $\alpha \in \mathcal{V}_{\mathsf{ty}}$ or an applied type constructor $\kappa(\bar{\tau}_n)$ for some n-ary $\kappa \in \Sigma_{\mathsf{ty}}$ and types $\bar{\tau}_n$. To indicate that an expression e has type τ, we write $e : \tau$.

Preprint submitted to
Logical Methods in Computer Science

1.1.2. *Lambda-Preterms and Lambda-Terms.* To define our logic's terms, for a given type signature Σ_{ty}, we fix a set $\mathcal{V}$ of variables with associated types. We write $x\langle\tau\rangle$ for a variable named x with associated type τ. We require that $\mathcal{V}$ contains infinitely many variables of any type.

A *term signature* Σ is a set of constants. Each constant is associated with a type declaration of the form $\Pi\bar{\alpha}_m.\ \bar{\tau}_n \Rightarrow \upsilon$, where $\bar{\tau}_n$ and υ are types and $\bar{\alpha}_m$ is a tuple of distinct variables that contains all type variables from $\bar{\tau}_n$ and υ. The types $\bar{\tau}_n$ are the types of the parameters of the constant, and υ may be a function type if the constant takes nonparameter arguments. We require that Σ contains the logical symbols $\mathsf{T}, \mathsf{\bot} : o$; $\mathsf{\neg} : o \to o$; $\mathsf{\wedge}, \mathsf{\vee}, \mathsf{\rightarrow} : o \to o \to o$; and $\mathsf{\approx}, \mathsf{\not\approx} : \Pi\alpha.\ \alpha \to \alpha \to o$. A type signature and a term signature form a *signature*.

Our syntax makes use of a locally nameless notation [10] using De Bruijn indices [11]. We distinguish between λ-preterms, λ-terms, preterms, and terms. Roughly, λ-preterms are raw syntactic expressions, λ-terms are the subset of locally closed λ-preterms, preterms are $\beta\eta$-equivalence classes of λ-preterms, and terms are $\beta\eta$-equivalence classes of λ-terms. More precisely, we define these notions as follows.

The set of λ-preterms is built from the following expressions:

- a variable $x\langle\tau\rangle : \tau$ for $x\langle\tau\rangle \in \mathcal{V}$;
- a symbol $\mathsf{f}\langle\bar{\upsilon}_m\rangle(\bar{u}_n) : \tau$ for a constant $\mathsf{f} \in \Sigma$ with type declaration $\Pi\bar{\alpha}_m.\ \bar{\tau}_n \Rightarrow \tau$, types $\bar{\upsilon}_m$, and λ-preterms $\bar{u} : \bar{\tau}_n$ such that all De Bruijn indices in $\bar{u}$ are bound;
- a De Bruijn index $n\langle\tau\rangle : \tau$ for a natural number $n \geq 0$ and a type τ, where τ represents the type of the bound variable;
- a λ-expression $\lambda\langle\tau\rangle\ t : \tau \to \upsilon$ for a type τ and a λ-preterm $t : \upsilon$ such that all De Bruijn indices bound by the new $\lambda\langle\tau\rangle$ have type τ;
- an application $s\ t : \upsilon$ for λ-preterms $s : \tau \to \upsilon$ and $t : \tau$.

The type arguments $\langle\bar{\tau}\rangle$ carry enough information to enable typing of any λ-preterm without any context. We often leave them implicit, when they are irrelevant or can be inferred. In $\mathsf{f}\langle\bar{\upsilon}_m\rangle(\bar{u}_n) : \tau$, we call $\bar{u}_n$ the parameters. We omit () when a symbol has no parameters. Notice that it is possible for a term to contain multiple occurrences of the same free De Bruijn index with different types. In contrast, the types of bound De Bruijn indices always match.

The set of λ-*terms* is the subset λ-preterms without free De Bruijn indices, i.e, the subset of locally closed λ-preterms. We write $\mathcal{T}^{\lambda}(\Sigma, \mathcal{V})$ for the set of all λ-terms and $\mathcal{T}^{\lambda\mathrm{pre}}(\Sigma, \mathcal{V})$ for the set of all λ-preterms, sometimes omitting the set $\mathcal{V}$ when it is clear from the context.

A λ-preterm is called *functional* if its type is of the form $\tau \to \upsilon$ for some types τ and υ. It is called *nonfunctional* otherwise.

Given a λ-preterm t and λ-terms $s_0, \ldots, s_n$, we write $t\{0 \mapsto s_0, \ldots, n \mapsto s_n\}$ for the λ-preterm resulting from substituting s_i for each De Bruijn index $i + j$ enclosed into exactly j λ-abstractions in t. For example, $(\mathsf{f}\ 0\ 1\ (\lambda\ \mathsf{g}\ 1\ 2))\{0 \mapsto \mathsf{a}, 1 \mapsto \mathsf{b}\} = \mathsf{f}\ \mathsf{a}\ \mathsf{b}\ (\lambda\ \mathsf{g}\ \mathsf{a}\ \mathsf{b})$. Given a λ-preterm t and a tuple $\bar{s}_n$ of λ-terms, we abbreviate $t\{0 \mapsto s_1, \ldots, (n-1) \mapsto s_n\}$ as $t\{(0, \ldots, n-1) \mapsto \bar{s}_n\}$.

We write $t{\downarrow_\beta}$ for the β-normal form of a λ-preterm t.

A λ-preterm s is a *subterm* of a λ-preterm t, written $t = t[s]$, if $t = s$, if $t = \mathsf{f}\langle\bar{\tau}_m\rangle(\bar{u})\ v$ with $u_i = u_i[s]$ or $v = v[s]$, if $t = \lambda\, u[s]$, if $t = (u[s])\ v$, or if $t = u\ (v[s])$. A subterm is *proper* if it is distinct from the λ-preterm itself.

A λ-preterm is *ground* if it contains no type variables and no term variables, i.e., if it is closed and monomorphic. We write $\mathcal{T}^{\lambda\text{pre}}_{\text{ground}}(\Sigma)$ for the set of ground λ-preterms and $\mathcal{T}^{\lambda}_{\text{ground}}(\Sigma)$ for the set of ground λ-terms.

1.1.3. *Preterms and Terms.* The set of (*pre*)*terms* consists of the $\beta\eta$-equivalence classes of λ-(pre)terms. For a given set of variables $\mathcal{V}$ and signature Σ, we write $\mathcal{T}(\Sigma, \mathcal{V})$ for the set of all terms and $\mathcal{T}^{\text{pre}}(\Sigma, \mathcal{V})$ for the set of all preterms, sometimes omitting the set $\mathcal{V}$ when it is clear from the context. We write $\mathcal{T}_{\text{ground}}(\Sigma)$ for the set of ground terms.

When referring to properties of a preterm that depend on the representative of its equivalence class modulo β (e.g., when checking whether a preterm is ground or whether a preterm contains a given variable x), we use a β-normal representative as the default representative of the $\beta\eta$-equivalence class. When referring to properties of a preterm that depend on the choice of representative modulo η, we state the intended representative explicitly.

Clearly, any preterm in β-normal form has one of the following four mutually exclusive forms:

- $x\langle\tau\rangle\ \bar{t}$ for a variable $x\langle\tau\rangle$ and terms $\bar{t}$;
- $\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\ \bar{t}$ for a symbol f, types $\bar{\tau}$, and terms $\bar{u}$, $\bar{t}$;
- $n\langle\tau\rangle\ \bar{t}$ for a De Bruijn index $n\langle\tau\rangle$ and terms $\bar{t}$;
- $\lambda\langle\tau\rangle\ t$ for a term t.

1.1.4. *Substitutions.* A substitution is a mapping ρ from type variables $\alpha \in \mathcal{V}_{\mathsf{ty}}$ to types $\alpha\rho$ and from term variables $x\langle\tau\rangle \in \mathcal{V}$ to (λ-)terms $x\rho : \tau\rho$. A substitution ρ applied to a (λ-)term t yields a (λ-)term $t\rho$ in which each variable x is replaced by $x\rho$. Similarly, subsitutions can be applied to types. The notation $\{\bar{\alpha} \mapsto \bar{\tau}, \bar{x} \mapsto \bar{t}\}$ denotes a substitution that maps each α_i to τ_i and each x_i to t_i, and all other type and term variables to themselves. The composition $\rho\sigma$ of two substitutions applies first ρ and then σ: $t\rho\sigma = (t\rho)\sigma$. A *grounding* substitution maps all variables to ground types and ground (λ-)terms. The notation $\sigma[\bar{x} \mapsto \bar{t}]$ denotes the substitution that maps each x_i to t_i and otherwise coincides with σ.

1.1.5. *Clauses.* Finally, we define the higher-order clauses on which our calculus operates. A *literal* is an unordered pair of two terms s and t associated with a positive or negative sign. We write positive literals as $s \approx t$ and negative literals as $s \not\approx t$. The notation $s \mathrel{\dot{\approx}} t$ stands for either $s \approx t$ or $s \not\approx t$. Nonequational literals are not supported and must be encoded as $s \approx \mathsf{T}$ or $s \approx \bot\!\!\!\bot$. A clause $L_1 \vee \cdots \vee L_n$ is a finite multiset of literals. The empty clause is written as $\bot$. Finally, we define a grounding function $\mathcal{G}$ on clauses as $\mathcal{G}(C) = \{C\theta \mid \theta \text{ is a grounding substitution}\}$

1.2. Semantics. The semantics is essentially the same as in Bentkamp et al. [5], adapted to the modified syntax.

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_{ty} 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_{ty} is recursively defined by $[\![\alpha]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}} = \xi_{\mathsf{ty}}(\alpha)$ and $[\![\kappa(\bar{\tau})]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}} = \mathcal{J}_{\mathsf{ty}}(\kappa)([\![\bar{\tau}]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}})$.

Given a type interpretation $\mathcal{I}_{\mathsf{ty}}$ and a type valuation ξ_{ty}, a *term valuation* ξ_{te} assigns an element $\xi_{\mathsf{te}}(x) \in [\![\tau]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}$ to each variable $x : \tau$. A valuation $\xi = (\xi_{\mathsf{ty}}, \xi_{\mathsf{te}})$ is a pair of a type valuation ξ_{ty} and a term valuation ξ_{te}.

An *interpretation function* $\mathcal{J}$ for a type interpretation $\mathcal{I}_{\mathsf{ty}}$ associates with each symbol $\mathsf{f} : \Pi\bar{\alpha}_m.\ \bar{\tau} \Rightarrow \upsilon$, a domain tuple $\bar{\mathcal{D}}_m \in \mathcal{U}^m$, and values $\bar{a} \in [\![\bar{\tau}]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}$ a value $\mathcal{J}(\mathsf{f}, \bar{\mathcal{D}}_m, \bar{a}) \in [\![\upsilon]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}$, where ξ_{ty} is a type valuation that maps each α_i to $\mathcal{D}_i$. We require that

(I1) $\mathcal{J}(\mathsf{T}) = 1$
(I2) $\mathcal{J}(\mathsf{\bot}) = 0$
(I3) $\mathcal{J}(\mathsf{\wedge})(a, b) = \min\{a, b\}$
(I4) $\mathcal{J}(\mathsf{\vee})(a, b) = \max\{a, b\}$
(I5) $\mathcal{J}(\mathsf{\neg})(a) = 1 - a$
(I6) $\mathcal{J}(\mathsf{\rightarrow})(a, b) = \max\{1 - a, b\}$
(I7) $\mathcal{J}(\mathsf{\approx}, \mathcal{D})(c, d) = 1$ if $c = d$ and 0 otherwise
(I8) $\mathcal{J}(\mathsf{\not\approx}, \mathcal{D})(c, d) = 0$ if $c = d$ and 1 otherwise

for all $a, b \in \{0, 1\}$, $\mathcal{D} \in \mathcal{U}$, and $c, d \in \mathcal{D}$.

The comprehension principle states that every function designated by a λ-expression is contained in the corresponding domain. Loosely following Fitting [13, Sect. 2.4], we initially allow λ-expressions to designate arbitrary elements of the domain, to be able to define the denotation of a λ-term. We impose restrictions afterward 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 $[\![\tau]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}$. We require that the value $\mathcal{L}(\xi, t)$ depends only on values of ξ at type and term variables that actually occur in t. 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 λ-term is defined as $[\![x]\!]^{\xi}_{\mathcal{I}} = \xi_{\mathsf{te}}(x)$, $[\![\mathsf{f}\langle\bar{\tau}\rangle(\bar{s})]\!]^{\xi}_{\mathcal{I}} = \mathcal{J}(\mathsf{f}, [\![\bar{\tau}]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}, [\![\bar{s}]\!]^{\xi}_{\mathcal{I}})$, $[\![s\ t]\!]^{\xi}_{\mathcal{I}} = [\![s]\!]^{\xi}_{\mathcal{I}}([\![t]\!]^{\xi}_{\mathcal{I}})$, and $[\![\lambda\langle\tau\rangle\ t]\!]^{\xi}_{\mathcal{I}} = \mathcal{L}(\xi, \lambda\langle\tau\rangle\ t)$. For ground λ-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 $[\![\lambda\langle\tau\rangle\ t]\!]^{(\xi_{\mathsf{ty}}, \xi_{\mathsf{te}})}_{\mathcal{I}}(a) = [\![t\{0 \mapsto x\}]\!]^{(\xi_{\mathsf{ty}}, \xi_{\mathsf{te}}[x \mapsto a])}_{\mathcal{I}}$ for all λ-expressions $\lambda\langle\tau\rangle\ t$ and all valuations ξ, where x is a fresh variable. Given an interpretation $\mathcal{I}$ and a valuation ξ, a positive literal $s \approx t$ (resp. negative literal $s \not\approx t$) is true if $[\![s]\!]^{\xi}_{\mathcal{I}}$ and $[\![t]\!]^{\xi}_{\mathcal{I}}$ are equal (resp. different). A clause is true if at least one of its literals is true. A set of clauses is true if all its elements are true. A proper interpretation $\mathcal{I}$ is a *model* of a set N of clauses, written $\mathcal{I} \models N$, if N is true in $\mathcal{I}$ for all valuations ξ. Given two sets M, N of clauses, we say that M *entails* N, written $M \models N$, if every model of M is also a model of N.

1.3. The Extensionality Skolem Constant. Any given signature can be extended with a distinguished constant $\mathsf{diff} : \Pi\alpha, \beta.\ (\alpha \to \beta, \alpha \to \beta) \Rightarrow \alpha$, which we require for our calculus. Interpretations as defined above can interpret the constant diff arbitrarily. The intended interpretation of diff is as follows:

Definition 1.1. We call a proper interpretation $\mathcal{I}$ diff-*aware* if $\mathcal{I}$ is a model of the extensionality axiom—i.e.,

$$\mathcal{I} \models z\ (\mathsf{diff}\langle\alpha, \beta\rangle(z, y)) \not\approx y\ (\mathsf{diff}\langle\alpha, \beta\rangle(z, y)) \vee z \approx y$$

Given two sets M, N of clauses, we write $M \models\!\!\approx N$ if every diff-aware interpretation that is a model of M is also a model of N.

Our calculus is sound and refutationally complete w.r.t. $\not\approx$ but unsound w.r.t. $\not\models$.

2. Calculus

2.1. Orange, Yellow, and Green Subterms. As in the original λ-superposition calculus, a central notion of our calculus is the notion of green subterms. These are the subterms that we consider for superposition inferences. For example, in the clause $\mathsf{f}\,\mathsf{a} \not\approx \mathsf{b}$, a superposition inference at a or $\mathsf{f}\,\mathsf{a}$ is possible, but not at f. Our definition here deviates from Bentkamp et al. [5] in that functional terms never have nontrivial green subterms.

In addition to green subterms, we define yellow subterms, which extend green subterms with subterms inside λ-expressions, and orange subterms, which extend yellow subterms with subterms containing free De Bruijn indices. Orange subterms are the subterms that our redundancy criterion allows simplification rules to rewrite at. For example, the clauses $\lambda\,\mathsf{c} \not\approx \mathsf{b}$ and $\mathsf{f}\,x\,x \approx \mathsf{c}$ can make $\lambda\,\mathsf{f}\,0\,0 \not\approx \mathsf{b}$ redundant (assuming a suitable clause order), but $\mathsf{g}\,\mathsf{a} \not\approx \mathsf{b}$ and $\mathsf{g} \approx \mathsf{f}$ cannot make $\mathsf{f}\,\mathsf{a} \not\approx \mathsf{b}$ redundant. It is convenient to define orange subterms first, then derive yellow and green subterms based on orange subterms.

Orange subterms depend on the choice of $\beta\eta$-normal form:

Definition 2.1 ($\beta\eta$-Normalizer)**.** Given a preterm t, let $t\downarrow_{\beta\eta\text{long}}$ be its β-normal η-long form and let $t\downarrow_{\beta\eta\text{short}}$ be its β-normal η-short form. A $\beta\eta$-normalizer is a function $\downarrow_{\beta\eta} \in \{\downarrow_{\beta\eta\text{long}}, \downarrow_{\beta\eta\text{short}}\}$.

Definition 2.2 (Orange Subterms)**.** We start by defining orange positions and orange subterms on λ-preterms.

Given a list of natural numbers p and $s, t \in \mathcal{T}^{\lambda\text{pre}}(\Sigma)$, we say that p is an *orange position* of t, and s is an *orange subterm* of t at p, written $t|_p = s$, if this can be derived inductively from the following rules:

1. $u|_\varepsilon = u$ for all $u \in \mathcal{T}^{\lambda\text{pre}}(\Sigma)$, where ε is the empty list.
2. If $u_i|_p = v$, then $(\mathsf{f}\langle\bar{\tau}\rangle(\bar{s})\;\bar{u}_n)|_{i.p} = v$ for all $\mathsf{f} \in \Sigma$, types $\bar{\tau}$, λ-preterms $\bar{s}, \bar{u}_n, v \in \mathcal{T}^{\lambda\text{pre}}(\Sigma)$, and $1 \le i \le n$.
3. If $u_i|_p = v$, then $(m\langle\tau\rangle\;\bar{u}_n)|_{i.p} = v$ for all De Bruijn indices m, types τ, λ-preterms $\bar{u}_n, v \in \mathcal{T}^{\lambda\text{pre}}(\Sigma)$, and $1 \le i \le n$.
4. If $u|_p = v$, then $(\lambda\langle\tau\rangle\;u)|_{1.p} = v$ for all types τ and λ-preterms $u, v \in \mathcal{T}^{\lambda\text{pre}}(\Sigma)$.

We extend these notions to preterms as follows. Given a $\beta\eta$-normalizer $\downarrow_{\beta\eta}$, a list of natural numbers p and $s, t \in \mathcal{T}^{\text{pre}}(\Sigma)$, we say that p is an *orange position* of t, and s is an *orange subterm* of t at p w.r.t. $\downarrow_{\beta\eta}$, written $t|_p = s$, if $(t\downarrow_{\beta\eta})|_p = s\downarrow_{\beta\eta}$.

The context $u[\,]$ surrounding an orange subterm s of $u[s]$ is called an *orange context*. The notation $u\llparenthesis s\rrparenthesis_p$ or $u\llparenthesis s\rrparenthesis$ indicates that s is an orange subterm in $u[s]$ at position p, and $u\llparenthesis\,\rrparenthesis$ indicates that $u[\,]$ is an orange context.

Example 2.3. Whether a preterm is an orange subterm of another preterm depends on the chosen $\beta\eta$-normal form $\downarrow_{\beta\eta}$. For example, the preterms $\mathsf{f}\,0$ and 0 are orange subterms of $\lambda\,\mathsf{f}\,0$ in η-long form, but they are not orange subterms of the η-short form f of the same term.

Remark 2.4. The possible reasons for a subterm not to be orange are the following:

– It is applied to arguments.
– It occurs inside a parameter.
– It occurs inside an argument of an applied variable.

Definition 2.5 (Yellow Subterms)**.** Let $\downarrow_{\beta\eta}$ be a $\beta\eta$-normalizer. A *yellow subterm* w.r.t. $\downarrow_{\beta\eta}$ is an orange subterm that does not contain free De Bruijn indices. A *yellow position* w.r.t. $\downarrow_{\beta\eta}$ is an orange position that identifies a yellow subterm. The context surrounding a yellow subterm is called a *yellow context*.

Lemma 2.6. *Whether a preterm is a yellow subterm of another preterm is independent of* $\downarrow_{\beta\eta}$*. (On the other hand, its yellow position may differ.)*

Proof. It suffices to show that a single η-expansion or η-contraction from a β-reduced λ-preterm s into another β-reduced λ-preterm cannot remove yellow subterms. This suffices because only such η-conversations are needed to transform a β-normal η-long form into a β-normal η-short form and vice versa.

Assume s has a yellow subterm at yellow position p. Consider the possible forms that a β-reduced λ-preterm s can have:

- $x\langle\tau\rangle\,\bar{t}$ for a variable $x\langle\tau\rangle$ and λ-preterms $\bar{t}$;
- $\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\,\bar{t}$ for a symbol f, types $\bar{\tau}$, and λ-preterms $\bar{u}$, $\bar{t}$;
- $n\langle\tau\rangle\,\bar{t}$ for a De Bruijn index $n\langle\tau\rangle$ and λ-preterms $\bar{t}$;
- $\lambda\langle\tau\rangle\,t$ for a λ-preterm t.

Consider where an η-conversion could happen: If an η-expansion takes place at the left-hand side of an application, the result is not β-reduced. If an η-reduction takes place at the left-hand side of an application, the original λ-preterm is not β-reduced. If the yellow subterm at p does not overlap with the place of η-conversion, the η-conversion has no effect on the yellow subterm. This excludes the case where the η-conversion takes place in an argument of an applied variable or in a parameter. So the only relevant subterms for η-conversions are (a) the entire λ-preterm s, (b) a subterm of $\bar{t}$ in $\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\,\bar{t}$, (c) a subterm of $\bar{t}$ in $n\langle\tau\rangle\,\bar{t}$, or (d) a subterm of t in $\lambda\langle\tau\rangle\,t$.

Next, we consider the possible positions p. If the η-conversion takes place inside of the yellow subterm, it certainly remains orange because orange subterms only depend on the outer structure of the λ-preterm. It also remains yellow because η-conversion does not introduce free De Bruijn indices. This covers in particular the case where p is the empty list. Otherwise, the yellow subterm at p is also (i) a yellow subterm of $\bar{t}$ in $\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\,\bar{t}$, (ii) a yellow subterm of $\bar{t}$ in $n\langle\tau\rangle\,\bar{t}$, or (iii) a yellow subterm of t in $\lambda\langle\tau\rangle\,t$. In cases (b), (c), and (d), we can apply the induction hypothesis to $\bar{t}$ or t and conclude that the yellow subterm of $\bar{t}$ or t remains yellow and thus the yellow subterm of s at p remains yellow as well. In case (a), we distinguish between the cases (i) to (iii) described above:

(i) Then the only option is an η-expansion of $\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\,\bar{t}$ to $\lambda\,\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\,\bar{t}\,0$. Clearly, the yellow subterm in $\bar{t}$ remains yellow, although its yellow position changes.

(ii) Analogous to (i).

(iii) Here, one option is an η-expansion of $\lambda\,t$ to $\lambda\,\lambda\,t\,0$, which can be treated analogously to (i).

The other option is an η-reduction of $\lambda\,t$ to t', where $t = t'\,0$. We must show that a yellow subterm of t is also a yellow subterm of t'. Since a yellow subterm of t cannot contain the free De Bruijn index 0, the λ-preterm t' must be of the form $v\,\bar{w}$, where the preterm v is a symbol or a De Bruijn index and the yellow subterm of $t = v\,\bar{w}\,0$ must be a yellow subterm of one of the arguments $\bar{w}$. Then it is also a yellow subterm of $v\,\bar{w} = t'$. □

Definition 2.7 (Green Subterms)**.** A *green position* is an orange position p such that each orange subterm at a proper prefix of p is nonfunctional. *Green subterms* are orange subterms at green positions. The context surrounding a green subterm s of $u[s]$ is called a *green context*. The notation $u\langle s\rangle_p$ or $u\langle s\rangle$ indicates that s is a green subterm in $u[s]$ at position p, and $u\langle\,\rangle$ indicates that $u[\,]$ is a green context.

Clearly, green subterms can equivalently be described as follows: Every term is a green subterm of itself. If u is nonfunctional, then every green subterm of one of its arguments s_i is a green subterm of $u = \mathsf{f}(\bar{t})\;\bar{s}$ and of $u = n\;\bar{t}$. Moreover, since η-conversions can occur only at functional subterms, both green subterms and green positions do not depend on the choice of a $\beta\eta$-normalizer $\downarrow_{\beta\eta}$.

Example 2.8. Let ι be a type constructor. Let α be a type variable. Let $x : \iota \to \iota$ be a variable. Let $\mathsf{a} : \iota$, $\mathsf{f} : \Pi\alpha.\, \iota \Rightarrow (\iota \to \iota) \to \alpha$, and $\mathsf{g} : \iota \to \iota \to \iota$ be constants. Consider the term $\mathsf{f}\langle\alpha\rangle(\mathsf{a})\,(\lambda\,\mathsf{g}\,(x\,\mathsf{a})\,0)$. Its green subterms are the entire term (at position ε) and $\lambda\,\mathsf{g}\,(x\,\mathsf{a})\,0$ (at position 1). Its yellow subterms are the green subterms and $x\,\mathsf{a}$ (at position 1.1.1 w.r.t. $\downarrow_{\beta\eta\text{long}}$ or at position 1.1 w.r.t. $\downarrow_{\beta\eta\text{short}}$). Its orange subterms w.r.t. $\downarrow_{\beta\eta\text{long}}$ are the yellow subterms and $\mathsf{g}\,(x\,\mathsf{a})\,0$ (at position 1.1) and 0 (at position 1.1.2). Using $\downarrow_{\beta\eta\text{short}}$, the orange subterms of this term are exactly the yellow subterms.

For positions in clauses, natural numbers are not appropriate because clauses and literals are unordered. A solution is the following definition:

Definition 2.9 (Orange, Yellow, and Green Positions and Subterms in Clauses)**.** Let C be a clause, let $L = s \mathrel{\dot{\approx}} t$ be a literal in C, and let p be an orange position of s. Then we call the expression $L.s.p$ an *orange position* in C, and the *orange subterm* of C at position $L.s.p$ is the orange subterm of s at position p. *Yellow positions/subterms* and *green positions/subterms* of clauses are defined analogously.

Example 2.10. The clause $C = K \vee L$ with $K = \mathsf{f}\;\mathsf{a} \not\approx \mathsf{b}$ and $L = \mathsf{c} \approx \mathsf{f}\;\mathsf{a}$ contains the orange subterm a twice, once at orange position $L.(\mathsf{f}\;\mathsf{a}).1$ and once at orange position $K.(\mathsf{f}\;\mathsf{a}).1$.

2.2. Complete Sets of Unifiers.

Definition 2.11. Given a set of constraints S and a set X of variables, where X contains at least the variables occurring in S, a *complete set of unifiers* is a set P of unifiers of S such that for each unifier θ of S, there exists a substitution $\sigma \in P$ and a substitution ρ such that $x\sigma\rho = x\theta$ for all $x \in X$.

Given a set of constraints S and a set X of variables, we write $\mathrm{CSU}_X(S)$ or $\mathrm{CSU}(S)$ for an arbitrary complete set of unifiers. Again, we require that all elements of $\mathrm{CSU}(S)$ unify at least the types of the terms pairs in S and that all elements of $\mathrm{CSU}(S)$ are idempotent.

2.3. Term Orders and Selection Functions.

Our calculus is parameterized by a relation $\succ$ on terms, literals, and clauses. We call $\succ$ the term order, but it need not formally be a partial order. Moreover, our calculus is parameterized by a literal selection function.

The original λ-superposition calculus also used a nonstrict term order $\succsim$ to compare terms that may become equal when instatiated, such as $x\;\mathsf{b} \succsim x\;\mathsf{a}$, where $\mathsf{b} \succ \mathsf{a}$. However,

contrary to the claims made for the original λ-superposition calculus, employing the nonstrict term order can lead to incompleteness [6], which is why we do not use it in our calculus.

Moreover, the original λ-superposition calculus used a Boolean selection function to restrict inferences on clauses containing Boolean subterms. For simplicity, we omit this feature in our calculus because an evaluation did not reveal any practical benefit [18].

Definition 2.12 (Admissible Term Order)**.** A relation $\succ$ on terms and on clauses is an *admissible term order* if it fulfills the following criteria, where $\succeq$ denotes the reflexive closure of $\succ$:

(O1) the relation $\succ$ on ground terms is a well-founded total order;
(O2) ground compatibility with yellow contexts: $s' \succ s$ implies $t\langle\!\langle s' \rangle\!\rangle \succ t\langle\!\langle s \rangle\!\rangle$ for ground terms s, s', and t;
(O3) ground yellow subterm property: $t\langle\!\langle s \rangle\!\rangle \succeq s$ for ground terms s and t;
(O4) $u \succ \bot \succ \top$ for all ground terms $u \notin \{\top, \bot\}$;
(O5) $u \succ u\,\mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$ for all ground types τ, υ and ground terms $s, t, u : \tau \to \upsilon$;
(O6) the relation $\succ$ on ground clauses is the standard extension of $\succ$ on ground terms via multisets [1, Sect. 2.4];
(O7) stability under grounding substitutions for terms: $t \succ s$ implies $t\theta \succ s\theta$ for all grounding substitutions θ;
(O8) stability under grounding substitutions for clauses: $D \succ C$ implies $D\theta \succ C\theta$ for all grounding substitutions θ;
(O9) transitivity on literals: the relation $\succ$ on literals is transitive;

Definition 2.13 (Maximality)**.** Given a term order $\succ$, a literal K of a clause C is *maximal* if for all $L \in C$ such that $L \succeq K$, we have $L \preceq K$. It is *strictly* maximal if it is maximal and occurs only once in C.

In addition to the term order, our calculus is parameterized by a selection function:

Definition 2.14 (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*. Only negative literals and literals of the form $t \approx \bot$ may be selected.

Based on the term order and the selection function, we define *eligibility* as follows:

Definition 2.15 (Eligibility)**.** A literal L is (*strictly*) *eligible* w.r.t. a substitution σ in C if it is selected in C or there are no selected literals in C and $L\sigma$ is (strictly) maximal in $C\sigma$.

A green position $L.s.p$ of a clause C is *eligible* w.r.t. a substitution σ if the literal L is either negative and eligible or positive and strictly eligible (w.r.t. σ in C); and L is of the form $s \mathrel{\dot{\approx}} t \in C$ such that $s\sigma \not\preceq t\sigma$.

When we do not specify a substitution, we mean eligibility w.r.t. the identity substitution.

2.4. Concrete Term Orders. A companion article [3] defines two concrete term orders fulfilling the criteria of Definition 2.12: λKBO, inspired by the Knuth–Bendix order, and λLPO, inspired by the lexicographic path order. Since the companion article defines the orders only on terms, we extend $\succ_{\mathsf{\lambda kbo}}$ and $\succ_{\mathsf{\lambda lpo}}$ to literals and clauses via the standard extension using multisets [1, Sect. 2.4].

Theorem 2.16. *Let $\succ_{\mathsf{\lambda kbo}}$ denote the strict variant of λKBO as defined in the companion article. The order is parameterized by a precedence relation $>$ on symbols, a function w assigning weights to symbols, a constant w_{db} defining the weight of De Bruijn indices, and a function k assigning argument coefficients to symbols. Assume that these parameters fulfill $w(\mathsf{T}) = w(\mathsf{\bot}) = 1$, $w_{\mathsf{db}} \geq w(\mathsf{diff})$, $\mathsf{f} > \mathsf{\bot} > \mathsf{T}$ for all symbols $\mathsf{f} \notin \{\mathsf{T}, \mathsf{\bot}\}$, and $\mathsf{k}(\mathsf{diff}, i) = 1$ for every i. Using the extension defined above, $\succ_{\mathsf{\lambda kbo}}$ is an admissible term order.*

Proof. For most of the criteria, we use that by Theorems 4.11 and 5.11 of the companion article, $\succ_{\mathsf{g\lambda kbo}}$ is the restriction of $\succ_{\mathsf{\lambda kbo}}$ to ground terms.

(O1) By Theorems 3.8 and 3.10 of the companion article, $\succ_{\mathsf{g\lambda kbo}}$ is a total order. By Theorem 3.11 of the companion article, it is well founded.
(O2) By Theorem 3.14 of the companion article, $\succ_{\mathsf{g\lambda kbo}}$ is compatible with orange contexts and thus also with yellow contexts.
(O3) By Theorem 3.15 of the companion article, $\succ_{\mathsf{g\lambda kbo}}$ enjoys the orange subterm property and thus also the yellow subterm property.
(O4) By Theorem 3.16 of the companion article, $u \succ_{\mathsf{g\lambda kbo}} \mathsf{\bot} \succ_{\mathsf{g\lambda kbo}} \mathsf{T}$ for all ground terms $u \notin \{\mathsf{T}, \mathsf{\bot}\}$, using our assumptions about the weight and precedence of T and $\mathsf{\bot}$.
(O5) By Theorem 3.17 of the companion article, $u \succ_{\mathsf{g\lambda kbo}} u\,\mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$ for all ground types τ, υ and ground terms $s, t, u : \tau \to \upsilon$, using our assumptions about the weight and argument coefficients of diff.
(O6) By definition of our extension of $\succ_{\mathsf{\lambda kbo}}$ to clauses.
(O7) By Theorems 4.10 and 5.10 of the companion article.
(O8) Using the Dershowitz–Manna definition [12] of a multiset, it is easy to see that stability under substitutions for terms implies stability under substitutions for clauses.
(O9) By Theorem 5.13 of the companion article, $\succ_{\mathsf{\lambda kbo}}$ is transitive on terms. Since the multiset extension preserves transitivity, it is also transitive on literals.

□

Theorem 2.17. *Let $\succ_{\mathsf{\lambda lpo}}$ denote the strict variant of λLPO as defined in the companion article. The order is parameterized by a precedence relation $>$ on symbols and a watershed symbol ws. Assume that $\mathsf{f} > \mathsf{\bot} > \mathsf{T}$ for all symbols $\mathsf{f} \notin \{\mathsf{T}, \mathsf{\bot}\}$, that $\mathsf{\bot} \leq \mathsf{ws}$, and that $\mathsf{diff} \leq \mathsf{ws}$. Using the extension defined above, $\succ_{\mathsf{\lambda lpo}}$ is an admissible term order.*

Proof. For most of the criteria, we use that by Theorems 4.20 and 5.17 of the companion article, $\succ_{\mathsf{g\lambda lpo}}$ is the restriction of $\succ_{\mathsf{\lambda lpo}}$ to ground terms.

(O1) By Theorems 3.21 and 3.22 of the companion article, $\succ_{\mathsf{g\lambda lpo}}$ is a total order. By Theorem 3.23 of the companion article, it is well founded.
(O2) By Theorem 3.24 of the companion article, $\succ_{\mathsf{g\lambda lpo}}$ is compatible with orange contexts and thus also with yellow contexts.
(O3) By Theorem 3.25 of the companion article, $\succ_{\mathsf{g\lambda lpo}}$ enjoys the orange subterm property and thus also the yellow subterm property.
(O4) By Theorem 3.26 of the companion article, $u \succ_{\mathsf{g\lambda lpo}} \mathsf{\bot} \succ_{\mathsf{g\lambda lpo}} \mathsf{T}$ for all ground terms $u \notin \{\mathsf{T}, \mathsf{\bot}\}$, using our assumptions about the precedence of T and $\mathsf{\bot}$.
(O5) By Theorem 3.27 of the companion article, $u \succ_{\mathsf{g\lambda lpo}} u\,\mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$ for all ground types τ, υ and ground terms $s, t, u : \tau \to \upsilon$, using our assumption about the precedence of diff.
(O6) By definition of our extension of $\succ_{\mathsf{\lambda lpo}}$ to clauses.
(O7) By Theorems 4.19 and 5.16 of the companion article.

(O8) Using the Dershowitz–Manna definition [12] of a multiset, it is easy to see that stability under substitutions for terms implies stability under substitutions for clauses.
(O9) By Theorem 5.19 of the companion article, $\succ_{\mathsf{\lambda lpo}}$ is transitive on terms. Since the multiset extension preserves transitivity, it is also transitive on literals.

□

2.5. The Core Inference Rules. The calculus is parameterized by an admissible term order $\succ$ and a selection function *hsel*. We denote this calculus as $HInf^{\succ,hsel}$ or just *HInf*.

Each of our inference rules describes a collection of inferences, which we formally define as follows:

Definition 2.18. An *inference* ι is a tuple $(C_1, C_2, \ldots, C_{n+1})$ of clauses, written

$$\frac{C_1 \quad C_2 \quad \cdots \quad C_n}{C_{n+1}}$$

The clauses $C_1, C_2, \ldots, C_n$ are called *premises*, denoted by $prems(\iota)$, and C_{n+1} is called *conclusion*, denoted by $concl(\iota)$. The clause C_n is called the *main premise* of ι, denoted by $mprem(\iota)$. We assume that the premisses of an inference do not have any variables in common, which can be achieved by renaming them apart when necessary.

Our variant of the superposition rule, originating from the standard superposition calculus, is stated as follows:

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

1. $\sigma \in \mathrm{CSU}(t \equiv u)$;
2. u is not a variable, unless there exists another occurrence of that variable inside of a parameter in C;
3. $u\sigma$ is nonfunctional;
4. $t\sigma \not\preceq t'\sigma$;
5. the position of u is eligible in C w.r.t. σ;
6. $t \approx t'$ is strictly maximal in D w.r.t. σ;
7. there are no selected literals in D.

The rule FluidSup simulates superposition below applied variables:

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

with the following side conditions, in addition to Sup's conditions 3 to 7:

1. $\sigma \in \mathrm{CSU}(z\ t \equiv u)$;
2. u is variable-headed, and if u is a variable, then there exists another occurrence of that variable inside of a parameter in C;
8. z is a fresh variable;
9. $(z\ t)\sigma \neq (z\ t')\sigma$;
10. $z\sigma \neq \lambda\, 0$.

The equality resolution rule EqRes and the equality factoring rule EqFact also originate from the standard superposition calculus:

$$\frac{\overbrace{C' \vee u \not\approx u'}^{C}}{C'\sigma}\,\textsc{EqRes} \qquad\qquad \frac{\overbrace{C' \vee u' \approx v' \vee u \approx v}^{C}}{(C' \vee v \not\approx v' \vee u \approx v')\sigma}\,\textsc{EqFact}$$

Side conditions for EqRes:

1. $\sigma \in \mathrm{CSU}(u \equiv u')$;
2. $u \not\approx u'$ is eligible in C w.r.t. σ.

Side conditions for EqFact:

1. $\sigma \in \mathrm{CSU}(u \equiv u')$;
2. $u \approx v$ is eligible in C w.r.t. σ;
3. there are no selected literals in C;
4. $u\sigma \not\preceq v\sigma$.

The following rules Clausify, BoolHoist, LoobHoist, and FalseElim are responsible for converting Boolean terms into clausal form. The rules BoolHoist and LoobHoist each come with an analogue, respectively called FluidBoolHoist and FluidLoobHoist, which simulates their application below applied variables.

$$\frac{C' \vee s \approx t}{(C' \vee D)\sigma}\,\textsc{Clausify}$$

with the following side conditions:

1. $\sigma \in \mathrm{CSU}(s \equiv s', t \equiv t')$;
2. $s \approx t$ is strictly eligible in C w.r.t. σ;
3. s is not a variable, unless there exists an occurrence of that variable inside of a parameter in C;
4. the triple (s', t', D) is one of the following, where α is a fresh type variable and x and y are fresh term variables:

$(x \mathbin{\mathsf{\wedge}} y,\ \mathsf{T},\ x \approx \mathsf{T})$	$(x \mathbin{\mathsf{\wedge}} y,\ \mathsf{T},\ y \approx \mathsf{T})$	$(x \mathbin{\mathsf{\wedge}} y,\ \bot,\ x \approx \bot \vee y \approx \bot)$
$(x \mathbin{\mathsf{\vee}} y,\ \mathsf{T},\ x \approx \mathsf{T} \vee y \approx \mathsf{T})$	$(x \mathbin{\mathsf{\vee}} y,\ \bot,\ x \approx \bot)$	$(x \mathbin{\mathsf{\vee}} y,\ \bot,\ y \approx \bot)$
$(x \mathbin{\mathsf{\to}} y,\ \mathsf{T},\ x \approx \bot \vee y \approx \mathsf{T})$	$(x \mathbin{\mathsf{\to}} y,\ \bot,\ x \approx \mathsf{T})$	$(x \mathbin{\mathsf{\to}} y,\ \bot,\ y \approx \bot)$
$(x \mathbin{\mathsf{\approx}}\langle\alpha\rangle\, y,\ \mathsf{T},\ x \approx y)$	$(x \mathbin{\mathsf{\approx}}\langle\alpha\rangle\, y,\ \bot,\ x \not\approx y)$	
$(x \mathbin{\mathsf{\not\approx}}\langle\alpha\rangle\, y,\ \mathsf{T},\ x \not\approx y)$	$(x \mathbin{\mathsf{\not\approx}}\langle\alpha\rangle\, y,\ \bot,\ x \approx y)$	
$(\mathsf{\neg} x,\ \mathsf{T},\ x \approx \bot)$	$(\mathsf{\neg} x,\ \bot,\ x \approx \mathsf{T})$	

$$\frac{C\langle u\rangle}{(C\langle\bot\rangle \vee u \approx \mathsf{T})\sigma}\,\textsc{BoolHoist} \qquad\qquad \frac{C\langle u\rangle}{(C\langle\mathsf{T}\rangle \vee u \approx \bot)\sigma}\,\textsc{LoobHoist}$$

each with the following side conditions:

1. σ is the most general type substitution such that $u\sigma$ is of Boolean type (i.e., the identity if u is of Boolean type or $\{\alpha \mapsto o\}$ if u is of type α for some type variable α);
2. u is neither T nor $\bot$, and if u is a variable, there exists another occurrence of that variable inside of a parameter in C;
3. the position of u is eligible in C w.r.t. σ;

4. the occurrence of u is not in a literal of the form $u \approx \bot$ or $u \approx \top$.

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

1. u is variable-headed, and if u is a variable, there exists another occurrence of that variable inside of a parameter in C;
2. $u\sigma$ is nonfunctional;
3. x is a fresh variable of Boolean type, and z is a fresh variable of function type from Boolean to the type of u;
4. $\sigma \in \mathrm{CSU}(z\ x \equiv u)$;
5. $(z\ \bot)\sigma \neq (z\ x)\sigma$;
6. $z\sigma \neq \lambda\, 0$;
7. $x\sigma \neq \top$ and $x\sigma \neq \bot$;
8. the position of u is eligible in C w.r.t. σ.

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

with the same side conditions as FluidBoolHoist, but where $\bot$ is replaced by $\top$ in condition 5.

$$\frac{\overbrace{C' \vee s \approx t}^{C}}{C'\sigma}\ \textsc{FalseElim}$$

with the following side conditions:

1. $\sigma \in \mathrm{CSU}(s \equiv \bot, t \equiv \top)$;
2. $s \approx t$ is strictly eligible in C w.r.t. σ.

The argument congruence rule ArgCong and the extensionality rule Ext convert functional terms into nonfunctional terms. The rule Ext also comes with an analogue FluidExt, which simulates its application below applied variables.

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

with the following side conditions:

1. σ is the most general type substitution such that $s\sigma$ is functional (i.e., the identity if s is functional or $\{\alpha \mapsto (\beta \to \gamma)\}$ for fresh β and γ if s is of type α for some type variable α);
2. $s \approx s'$ is strictly eligible in C w.r.t. σ;
3. x is a fresh variable.

$$\frac{C\langle u\rangle}{C\sigma\langle y\rangle \vee u\sigma\ (\mathsf{diff}\langle\tau,\upsilon\rangle(u\sigma, y)) \not\approx y\ (\mathsf{diff}\langle\tau,\upsilon\rangle(u\sigma, y))}\ \textsc{Ext}$$

with the following side conditions:

1. σ is the most general type substitution such that $u\sigma$ is of type $\tau \to \upsilon$ for some τ and υ;
2. y is a fresh variable of the same type as $u\sigma$;
3. the position of u is eligible in C w.r.t. σ.

$$\frac{C\langle u\rangle}{(C\langle z\ y\rangle \vee x\ (\mathsf{diff}\langle\alpha,\beta\rangle(x,y)) \not\approx y\ (\mathsf{diff}\langle\alpha,\beta\rangle(x,y)))\sigma}\ \textsc{FluidExt}$$

with the following side conditions:

1. u is variable-headed;
2. $u\sigma$ is nonfunctional;
3. x and y are fresh variables of type $\alpha \to \beta$, and z is a fresh variable of function type from $\alpha \to \beta$ to the type of u;
4. $\sigma \in \mathrm{CSU}(S, z\ x \equiv u)$;
5. $(z\ x)\sigma \neq (z\ y)\sigma$;
6. $z\sigma \neq \lambda\, 0$;
7. the position of u is eligible in C w.r.t. σ.

Our calculus also includes the following axiom (i.e., nullary inference rule), which establishes the interpretation of the extensionality Skolem constant diff.

$$\frac{}{y\ (\mathsf{diff}\langle\alpha,\beta\rangle(y,z)) \not\approx z\ (\mathsf{diff}\langle\alpha,\beta\rangle(y,z)) \vee y\ x \approx z\ x}\ \textsc{Diff}$$

2.6. Redundancy. Our calculus includes a redundancy criterion that can be used to delete certain clauses and avoid certain inferences deemed redundant. The criterion is based on a translation to ground monomorphic first-order logic.

Let Σ be a higher-order signature. We require Σ to contain a symbol $\mathsf{diff} : \Pi\alpha,\beta.(\alpha \to \beta, \alpha \to \beta) \Rightarrow \alpha$. Based on this higher-order signature, we construct a first-order signature $\mathcal{F}(\Sigma)$ as follows. The type constructors are the same, but $\to$ is an uninterpreted symbol in the first-order logic. For each ground higher-order term of the form $\mathsf{f}\langle\bar{\tau}\rangle(\bar{u}) : \tau_1 \to \cdots \to \tau_m \to \tau$, with $m \geq 0$, we introduce a first-order symbol $\mathsf{f}^{\bar{\tau}}_{\bar{u}} : \tau_1 \times \cdots \times \tau_m \Rightarrow \tau$. Moreover, we introduce a first-order symbol $\mathsf{fun}_t : \tau \to \upsilon$ for each higher-order term t of type $\tau \to \upsilon$.

We define an encoding $\mathcal{F}$ from higher-order ground terms to first-order terms:

Definition 2.19. For ground terms t, we define $\mathcal{F}$ recursively as follows: If t is functional, then let $\mathcal{F}(t) = \mathsf{fun}_t$. Otherwise, t is of the form $\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})\ \bar{t}_m$, and we define $\mathcal{F}(t) = \mathsf{f}^{\bar{\tau}}_{\bar{u}}(\mathcal{F}(\bar{t}_1), \ldots, \mathcal{F}(\bar{t}_m))$.

For clauses, we apply $\mathcal{F}$ on each side of each literal individually.

Lemma 2.20. *The map $\mathcal{F}$ is a bijection between higher-order ground terms and first-order ground terms.*

Proof. We can see that $\mathcal{F}(s) = \mathcal{F}(t)$ implies $s = t$ for all ground s and t by structural induction on $\mathcal{F}(s)$. Moreover, we can show that for each first-order ground term t, there exists an s such that $\mathcal{F}(s) = t$ by structural induction on t. Injectivity and surjectivity imply bijectivity. □

We consider two different semantics for our first-order logic: $\models_{\mathrm{fol}}$ and $\models_{\mathrm{o}\lambda}$. The semantics $\models_{\mathrm{fol}}$ is the standard semantics of first-order logic. The semantics $\models_{\mathrm{o}\lambda}$ restricts $\models_{\mathrm{fol}}$ to interpretations $\mathcal{I}$ with the following properties:

– Interpreted Booleans: The domain of the Boolean type has exactly two elements, $[\![\mathsf{T}]\!]_\mathcal{I}$ and $[\![\bot]\!]_\mathcal{I}$, and the symbols $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\approx^\tau$, $\not\approx^\tau$ are interpreted as the corresponding logical operations.
– Extensionality w.r.t. diff: For all ground $u, w : \tau \to \upsilon$, if $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,t)) \approx \mathcal{F}(w\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,t))$ for all ground $s, t : \tau \to \upsilon$, then $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u) \approx \mathcal{F}(w)$.
– Argument congruence w.r.t. diff: For all ground $u, w, s, t : \tau \to \upsilon$, if $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u) \approx \mathcal{F}(w)$, then $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,t)) \approx \mathcal{F}(w\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,t))$.

2.6.1. *Clause Redundancy.* Our redundancy criterion for clauses provides two conditions that can make a clause redundant. The first condition applies when the ground instances of a clause are entailed by smaller ground instances of other clauses. It generalizes the standard superposition redundancy criterion to higher-order clauses. The second condition applies when there are other clauses with the same ground instances. It can be used to justify subsumption. For this second condition, we fix a well-founded partial order $\sqsupset$ on $\mathcal{C}_\text{H}$, which prevents infinite chains of clauses where each clause is made redundant by the next one. For example, following Bentkamp et al. [7, Sect. 3.4], a sensible choice is to define $C \sqsupset D$ if either C is larger than D in syntactic size (i.e., number of variables, constants, and De Bruijn indices), or if C and D have the same syntactic size and C contains fewer distinct variables than D.

Definition 2.21. Since $\mathcal{F}$ is bijective on ground terms by Lemma 2.20, we can convert a term order $\succ$ on higher-order terms into a relation $\succ_\mathcal{F}$ on ground first-order terms as follows. For two ground first-order terms s and t, let $s \succ_\mathcal{F} t$ if $\mathcal{F}^{-1}(s) \succ \mathcal{F}^{-1}(t)$.

Definition 2.22 (Clause Redundancy)**.** Given a clause C and a clause set N, let $C \in \mathit{HRed}_\text{C}(N)$ if for each grounding substitution θ at least one of the following two conditions holds:

1. $\{E \in \mathcal{F}(\mathcal{G}(N)) \mid E \prec_\mathcal{F} \mathcal{F}(C\theta)\} \models_{\text{o}\lambda} \mathcal{F}(C\theta)$; or
2. there exists a clause $D \in N$ and a grounding substitution ρ such that $C \sqsupset D$ and $D\rho = C\theta$.

2.6.2. *Inference Redundancy.* To define inference redundancy, we first define a calculus *FInf* on ground first-order logic with Booleans. It is parameterized by a relation $\succ$ on

ground first-order terms and a selection function on ground first-order clauses.

$$\frac{\overbrace{D' \vee t \approx t'}^{D} \quad C[t]}{D' \vee C[t']}\,\textsc{FSup} \qquad \frac{\overbrace{C' \vee u \not\approx u}^{C}}{C'}\,\textsc{FEqRes}$$

$$\frac{\overbrace{C' \vee u \approx v' \vee u \approx v}^{C}}{C' \vee v \not\approx v' \vee u \approx v}\,\textsc{FEqFact} \qquad \frac{C' \vee s \approx t}{C' \vee D}\,\textsc{FClausify}$$

$$\frac{C[u]}{(C[\bot] \vee u \approx \top)}\,\textsc{FBoolHoist} \qquad \frac{C[u]}{C[\top] \vee u \approx \bot}\,\textsc{FLoobHoist}$$

$$\frac{\overbrace{C' \vee \bot \approx \top}^{C}}{C'}\,\textsc{FFalseElim}$$

$$\frac{\overbrace{C' \vee \mathcal{F}(s) \approx \mathcal{F}(s')}^{C}}{C' \vee \mathcal{F}(s \operatorname{\mathsf{diff}}\langle \tau, \upsilon\rangle(u, w)) \approx \mathcal{F}(s' \operatorname{\mathsf{diff}}\langle \tau, \upsilon\rangle(u, w))}\,\textsc{FArgCong}$$

$$\frac{C[\mathcal{F}(u)]}{C[\mathcal{F}(w)] \vee \mathcal{F}(u \operatorname{\mathsf{diff}}\langle \tau, \upsilon\rangle(u, w)) \not\approx \mathcal{F}(w \operatorname{\mathsf{diff}}\langle \tau, \upsilon\rangle(u, w))}\,\textsc{FExt}$$

$$\frac{}{\mathcal{F}(u \operatorname{\mathsf{diff}}\langle \tau, \upsilon\rangle(u, w)) \not\approx \mathcal{F}(w \operatorname{\mathsf{diff}}\langle \tau, \upsilon\rangle(u, w)) \vee \mathcal{F}(u\ s) \approx \mathcal{F}(w\ s)}\,\textsc{FDiff}$$

Side conditions for FSup:

1. t is nonfunctional;
2. $t \succ t'$;
3. $D \prec C[t]$;
4. the position of t is eligible in C;
5. $t \approx t'$ is strictly eligible in D;
6. if t' is Boolean, then $t' = \top$.

Side conditions for FEqRes:

1. $u \not\approx u$ is eligible in C.

Side conditions for FEqFact:

1. $u \approx v$ is maximal in C;
2. there are no selected literals in C;
3. $u \succ v$,

Side conditions for FClausify:

1. $s \approx t$ is strictly eligible in $C' \vee s \approx t$;

2. The triple (s, t, D) has one of the following forms, where τ is an arbitrary type and u, v are arbitrary terms:

$(u \mathbin{\mathsf{\wedge}} v,\ \mathsf{T},\ u \approx \mathsf{T})$	$(u \mathbin{\mathsf{\wedge}} v,\ \mathsf{T},\ v \approx \mathsf{T})$	$(u \mathbin{\mathsf{\wedge}} v,\ \bot,\ u \approx \bot \vee v \approx \bot)$
$(u \mathbin{\mathsf{\vee}} v,\ \mathsf{T},\ u \approx \mathsf{T} \vee v \approx \mathsf{T})$	$(u \mathbin{\mathsf{\vee}} v,\ \bot,\ u \approx \bot)$	$(u \mathbin{\mathsf{\vee}} v,\ \bot,\ v \approx \bot)$
$(u \mathbin{\mathsf{\rightarrow}} v,\ \mathsf{T},\ u \approx \bot \vee v \approx \mathsf{T})$	$(u \mathbin{\mathsf{\rightarrow}} v,\ \bot,\ u \approx \mathsf{T})$	$(u \mathbin{\mathsf{\rightarrow}} v,\ \bot,\ v \approx \bot)$
$(u \approx^\tau v,\ \mathsf{T},\ u \approx v)$	$(u \approx^\tau v,\ \bot,\ u \not\approx v)$	
$(u \not\approx^\tau v,\ \mathsf{T},\ u \not\approx v)$	$(u \not\approx^\tau v,\ \bot,\ u \approx v)$	
$(\neg u,\ \mathsf{T},\ u \approx \bot)$	$(\neg u,\ \bot,\ u \approx \mathsf{T})$	

Side conditions for FBoolHoistand FLoobHoist:

1. u is of Boolean type
2. u is neither T nor $\bot$;
3. the position of u is eligible in C;
4. the occurrence of u is not in a literal L with $L = u \approx \bot$ or $L = u \approx \mathsf{T}$.

Side conditions for FFalseElim:

1. $\bot \approx \mathsf{T}$ is strictly eligible in C.

Side conditions for FArgCong:

1. s is of type $\tau \to \upsilon$;
2. u, w are ground terms of type $\tau \to \upsilon$;
3. $\mathcal{F}(s) \approx \mathcal{F}(s')$ is strictly eligible in C.

Side conditions for FExt:

1. the position of $\mathcal{F}(u)$ is eligible in C;
2. the type of u is $\tau \to \upsilon$;
3. w is a ground term of type $\tau \to \upsilon$;
4. $u \succ w$.

Side conditions for FDiff:

1. τ and υ are ground types;
2. u, w, s are ground terms.

Definition 2.23. We convert a selection function *hsel* on higher-order clauses into a selection function $\mathcal{F}(hsel)$ on ground first-order clauses as follows: Let a literal L of a first-order ground clause C be selected if $\mathcal{F}^{-1}(L)$ is selected in $\mathcal{F}^{-1}(C)$.

Definition 2.24. Let $\iota \in \mathit{HInf}^{\succ, hsel}$ for a term order $\succ$ and a selection function *hsel*. Let $C_1, \ldots, C_m$ be its premises and C_{m+1} its conclusion. Let $(\theta_1, \ldots, \theta_{m+1})$ be a tuple of grounding substitutions. We say that ι is *rooted* in *FInf* for $(\theta_1, \ldots, \theta_{m+1})$ if and only if

$$\frac{\mathcal{F}(C_1\theta_1) \quad \cdots \quad \mathcal{F}(C_m\theta_m)}{\mathcal{F}(C_{m+1}\theta_{m+1})}$$

is a valid $\mathit{FInf}^{\succ_\mathcal{F}, \mathcal{F}(hsel)}$ inference ι' such that the rule names of ι and ι' correspond up to the prefixes F and Fluid.

Definition 2.25 (Inference Redundancy)**.** Let $N \subseteq \mathcal{C}_\mathrm{H}$. Let $\iota \in \mathit{HInf}$ an inference with premises $C_1, \ldots, C_m$ and conclusion C_{m+1}. We define HRed_I so that $\iota \in \mathit{HRed}_\mathrm{I}(N)$ if for all substitutions $(\theta_1, \ldots, \theta_{m+1})$ for which ι is rooted in *FInf* (Definition 2.24), we have

– ι is a DIFF inference and $\mathcal{F}(\mathcal{G}(N)) \models_{\mathrm{o}\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$; or
– ι is some other inference and $\{E \in \mathcal{F}(\mathcal{G}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(C_m\theta_m)\} \models_{\mathrm{o}\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$.

2.7. Simplification Rules.

2.7.1. *Analogues of First-Order Simplification Rules.* Our notion of clause redundancy (Definition 2.22) can justify most analogues of the simplification rules implemented in Schulz's E prover [19, Sections 2.3.1 and 2.3.2]. Deletion of duplicated literals, deletion of resolved literals, syntactic tautology deletion, positive simplify-reflect, and negative simplify-reflect adhere to our redundancy criterion. Semantic tautology deletion can be applied as well, but we must use the entailment relation $\models_{\mathrm{o}\lambda}$ under the encoding $\mathcal{F}$.

Our analogue of clause subsumption is the following.

$$\frac{C \quad C\sigma \vee D}{C}\,\textsc{Subsumption}$$

with the following side condition:

1. $D \neq \bot$ or $C\sigma \sqsupset C$.

Lemma 2.26. SUBSUMPTION *can be justified by clause redundancy.*

Proof. Let θ be a grounding substitution. If D is nonempty, we apply condition 1 of Definition 2.22, which holds because the clause $C\sigma\theta$ is a proper subclause of $(C\sigma \vee D)\theta$ and therefore $\mathcal{F}(C\sigma\theta) \models_{\mathrm{o}\lambda} \mathcal{F}((C\sigma \vee D)\theta)$ and $C\sigma\theta \prec (C\sigma \vee D)\theta$. If $D = \bot$, we apply condition 2 of Definition 2.22, which holds by condition 1 of SUBSUMPTION. □

For rewriting of positive and negative literals (demodulation) and equality subsumption, we need to establish the following properties of orange subterms first:

Lemma 2.27. *Let $\downarrow_{\beta\eta}$ be a $\beta\eta$-normalizer. An orange subterm relation $u\langle\!\langle s\rangle\!\rangle_p$ w.r.t. $\downarrow_{\beta\eta}$ can be disassembled into a sequence $s_1 \ldots s_k$ as follows: s_1 is a green subterm of u; $s_k = s$; and for each $i < k$, $s_i = \lambda\, s'_i$ and s_{i+1} is a green subterm of s'_i.*

Proof. By induction on the size of u in η-long form.

If each orange subterm at a proper prefix of p is nonfunctional, then p is green, and we are done with $k = 1$ and $s_1 = s$.

Otherwise, let $p = q.r$ such that q is the shortest prefix with nonempty r, where the orange subterm s_1 at q is functional. Then s_1 is a green subterm of u at q because there does not exist a shorter prefix with a functional orange subterm. Moreover, since s_1 is functional, modulo η-conversion, $s_1 = \lambda\, s'_1$ for some s'_1. Since r is nonempty and s is the orange subterm of s_1 at r, there exists r' at most as long as r such that s is the orange subterm of s'_1 at r'. Specifically, if $s_1{\downarrow_{\beta\eta}}$ is a λ-abstraction, we use $1.r' = r$ and otherwise $r' = r$. By the induction hypothesis, since s is an orange subterm of s'_1, there exist $s_2, \ldots, s_k$ with $s_k = s$ such that $s_i = \lambda\, s'_i$ and s_{i+1} is a green subterm of s'_i for each $i < k$. □

Lemma 2.28. *Let $\downarrow_{\beta\eta}$ be a $\beta\eta$-normalizer. Let u be a ground term, and let p be an orange position of u w.r.t. $\downarrow_{\beta\eta}$. Let v, v' be ground preterms such that $u\langle\!\langle v\rangle\!\rangle_p$ and $u\langle\!\langle v'\rangle\!\rangle_p$ are terms.*

Let k be a number large enough such that $v\{(0,\dots,k-1)\mapsto \bar{t}_k\}$ and $v'\{(0,\dots,k-1)\mapsto \bar{t}_k\}$ do not contain free De Bruijn indices for all tuples of terms $\bar{t}_k$. Then

$$\{\mathcal{F}(v\{(0,\dots,k-1)\mapsto \bar{t}_k\} \approx v'\{(0,\dots,k-1)\mapsto \bar{t}_k\}) \mid \text{each } t_i \text{ of the form } \mathsf{diff}\langle \_,\_\rangle(\_,\_)\}$$
$$\models_{\mathrm{o}\lambda} \mathcal{F}(u\langle\!\langle v\rangle\!\rangle_p \approx u\langle\!\langle v'\rangle\!\rangle_p)$$

Proof. Let $\mathcal{I}$ be a $\models_{\mathrm{o}\lambda}$-interpretation with

$$\mathcal{I} \models_{\mathrm{o}\lambda} \mathcal{F}(v\{(0,\dots,k-1)\mapsto \bar{t}_k\} \approx v'\{(0,\dots,k-1)\mapsto \bar{t}_k\})$$

for all tuples of terms $\bar{t}_k$, where each t_i is of the form $\mathsf{diff}\langle \_,\_\rangle(\_,\_)$ for arbitrary values of '$\_$'. By Lemma 2.27, we have $u\langle\!\langle v\rangle\!\rangle_p = u\langle\lambda\, w_1\langle\lambda\, w_2\langle\cdots w_n\langle v\rangle\cdots\rangle\rangle\rangle$.

Step 1. Since v is a green subterm of $w_n\langle v\rangle$ and the terms $\bar{t}_k$ have a form that does not trigger β-reductions when substituting them for De Bruijn indices, $v\{(0,\dots,k-1)\mapsto \bar{t}_k\}$ is a green subterm of $w_n\langle v\rangle\{(0,\dots,k-1)\mapsto \bar{t}_k\}$ and thus

$$\mathcal{I} \models_{\mathrm{o}\lambda} \mathcal{F}(w_n\langle v\rangle\{(0,\dots,k-1)\mapsto \bar{t}_k\} \approx w_n\langle v'\rangle\{(0,\dots,k-1)\mapsto \bar{t}_k\})$$

Step 2. Using the property of extensionality w.r.t. diff of $\models_{\mathrm{o}\lambda}$-interpretations and using the fact that we have shown the above for all t_1 of the form $\mathsf{diff}\langle \_,\_\rangle(\_,\_)$, we obtain

$$\mathcal{I} \models_{\mathrm{o}\lambda} \mathcal{F}((\lambda\, w_n\langle v\rangle)\{0\mapsto t_2,\dots,(k-2)\mapsto t_k\} \approx (\lambda\, w_n\langle v'\rangle)\{0\mapsto t_2,\dots,(k-2)\mapsto t_k\})$$

Iterating steps 1 and 2 over $w_n,\dots,w_1,u$, we obtain

$$\mathcal{I} \models_{\mathrm{o}\lambda} \mathcal{F}(u\langle\!\langle v\rangle\!\rangle_p \approx u\langle\!\langle v'\rangle\!\rangle_p)$$ □

Our variant of rewriting of positive and negative literals (demodulation) is the following.

$$\frac{t \approx t' \quad C\langle\!\langle v\rangle\!\rangle}{t \approx t' \quad C\langle\!\langle v'\rangle\!\rangle}\ \textsc{Demod}$$

with the following side conditions:

1. $t\sigma = v\{(0,\dots,k-1)\mapsto \bar{x}_k\}$ and $t'\sigma = v'\{(0,\dots,k-1)\mapsto \bar{x}_k\}$ for some fresh variables $\bar{x}_k$ and a substitution σ.
2. $C\langle\!\langle v\rangle\!\rangle \succ C\langle\!\langle v'\rangle\!\rangle$;
3. for each tuple $\bar{t}_k$, where each t_i is of the form $\mathsf{diff}\langle \_,\_\rangle(\_,\_)$, we have $C\langle\!\langle v\rangle\!\rangle \succ v\{(0,\dots,k-1)\mapsto \bar{t}_k\} \approx v'\{(0,\dots,k-1)\mapsto \bar{t}_k\}$;

Remark 2.29. In general, it is unclear how to compute condition 3 of Demod. For λKBO and λLPO described in Section 2.4, however, the condition can easily be overapproximated by $C\langle\!\langle v\rangle\!\rangle \succ v \approx v'$, using the fact that the orders are also defined on preterms.

To prove that this is a valid overapproximation, it suffices to show the following: Let u and s be preterms with $u \succ s$ (resp. $u \succsim s$). Let s' be the result of replacing some De Bruijn indices in s by terms of the form $\mathsf{diff}\langle \_,\_\rangle(\_,\_)$. Then $u \succ s'$ (resp. $u \succsim s'$).

Proof for λKBO: By induction on the rule deriving $u \succ s$ or $u \succsim s$. Since we assume in Section 2.4 that $w_{\mathsf{db}} \geq w(\mathsf{diff})$ and $k(\mathsf{diff}, i) = 1$ for every i, we have $\mathcal{W}(s) \geq \mathcal{W}(s')$. It is easy to check that there is always a corresponding rule deriving $u \succ s'$ or $u \succsim s'$, in some cases using the induction hypothesis.

Proof for λLPO: By induction on the rule deriving $u \succ s$ or $u \succsim s$. Considering that we assume in Section 2.4 that $\mathsf{ws} \geq \mathsf{diff}$, it is easy to check that there is always a corresponding rule deriving $u \succ s'$ or $u \succsim s'$, in some cases using the induction hypothesis.

Since Demod makes use of orange subterms, it depends on the choice of $\beta\eta$-normalizer. Both $\downarrow_{\beta\eta\text{long}}$ and $\downarrow_{\beta\eta\text{short}}$ yield a valid simplification rule:

Lemma 2.30. Demod *can be justified by clause redundancy, regardless of the choice of $\beta\eta$-normalizer.*

Proof. Let θ be a grounding substitutiion. We apply condition 1 of Definition 2.22, using $C\langle\!\langle v\rangle\!\rangle$ for C. Let $T = \{\bar{t}_k \mid \text{each } t_i \text{ is a ground term of the form } \mathsf{diff}\langle \_, \_\rangle(\_, \_)\}$ and let $\rho_{\bar{t}_k} = \sigma\{\bar{x}_k \mapsto \bar{t}_k\}\theta$ for each $\bar{t}_k \in T$. By condition 1 of Demod,

$$\begin{aligned}(t \approx t')\rho_{\bar{t}_k} &= v\{(0, \ldots, k-1) \mapsto \bar{t}_k\}\theta \approx v'\{(0, \ldots, k-1) \mapsto \bar{t}_k\}\theta \\ &= v\theta\{(0, \ldots, k-1) \mapsto \bar{t}_k\} \approx v'\theta\{(0, \ldots, k-1) \mapsto \bar{t}_k\}\end{aligned}$$

for each tuple $\bar{t}_k \in T$.

By Lemma 2.28, $\mathcal{F}(\{(t \approx t')\rho_{\bar{t}_k} \mid \bar{t}_k \in T\}) \models_{\text{o}\lambda} \mathcal{F}(u\theta\langle\!\langle v\theta\rangle\!\rangle \approx u\theta\langle\!\langle v'\theta\rangle\!\rangle)$, where u is a side of a literal in $C\langle\!\langle v\rangle\!\rangle$ containing the orange subterm v. Thus

$$\mathcal{F}(\{(t \approx t')\rho_{\bar{t}_k} \mid \bar{t}_k \in T\}) \cup \{\mathcal{F}(C\langle\!\langle v'\rangle\!\rangle\theta)\} \models_{\text{o}\lambda} \mathcal{F}(C\langle\!\langle v\rangle\!\rangle\theta)$$

Condition 2 of Demod implies $C\langle\!\langle v'\rangle\!\rangle\theta \prec C\langle\!\langle v\rangle\!\rangle\theta$. Condition 3 of Demod implies $(t \approx t')\rho_{\bar{t}_k} \prec C\langle\!\langle v\rangle\!\rangle\theta$ for all $\bar{t}_k \in T$. Thus condition 1 of Definition 2.22 applies. □

Our variant of equality subsumption is the following:

$$\frac{t \approx t' \quad C' \vee s\langle\!\langle v\rangle\!\rangle \approx s'\langle\!\langle v'\rangle\!\rangle}{t \approx t'}\,\textsc{EqualitySubsumption}$$

with the following side conditions:

1. $t\sigma = v\{(0, \ldots, k-1) \mapsto \bar{x}_k\}$ and $t'\sigma = v'\{(0, \ldots, k-1) \mapsto \bar{x}_k\}$ for some fresh variables $\bar{x}_k$ and a substitution σ;
2. for each tuple $\bar{t}$, where each t_i is of the form $\mathsf{diff}\langle \_, \_\rangle(\_, \_)$, we have $C\langle\!\langle v\rangle\!\rangle \succ v\{(0, \ldots, k-1) \mapsto \bar{t}_k\} \approx v'\{(0, \ldots, k-1) \mapsto \bar{t}_k\}$;

To compute condition 2, we can exploit Remark 2.29.

Lemma 2.31. EqualitySubsumption *can be justified by simple clause redundancy, regardless of the choice of $\beta\eta$-normalizer.*

Proof. Analogous to Lemma 2.30. □

2.7.2. *Additional Simplification Rules.* The core inference rules ArgCong, Clausify, FalseElim, LoobHoist, and BoolHoist described in Section 2.5 can under certain conditions be applied as simplification rules.

Lemma 2.32. ArgCong *can be justified as a simplification rule by clause redundancy when σ is the identity. Moreover, it can even be applied when its eligibility condition does not hold.*

Proof. Let θ be a grounding substitution. We apply condition 1 of Definition 2.22. Let $\tau \to \upsilon$ be the type of $s\theta$ and $s'\theta$. By the extensionality property of $\models_{\text{o}\lambda}$, we have

$$\{\mathcal{F}((C' \vee s\,x \approx s'\,x)\theta[x \mapsto \mathsf{diff}\langle\tau, \upsilon\rangle(u, w)]) \mid u, w : \tau \to \upsilon \text{ ground}\} \models_{\text{o}\lambda} \mathcal{F}((C' \vee s \approx s')\theta)$$

By (O5), we have $(C' \vee s'\,x \approx s\,x)\theta[x \mapsto \mathsf{diff}\langle\tau, \upsilon\rangle(u, w)] \prec (C' \vee s \approx s')\theta$ for all such τ, υ, u, and w. Thus condition 1 of Definition 2.22 applies. □

Lemma 2.33. Clausify *can be justified as a simplification rule by clause redundancy when* σ *is the identity for all variables other than* x *and* y*. Moreover, it can even be applied when its eligibility condition does not hold.*

Proof. By condition 1 of Definition 2.22, using the fact that $\models_{o\lambda}$ interprets Booleans. □

Lemma 2.34. FalseElim *can be justified as a simplification rule by clause redundancy when* σ *is the identity. Moreover, it can even be applied when its eligibility condition does not hold.*

Proof. By condition 1 of Definition 2.22, using the fact that $\models_{o\lambda}$ interprets Booleans. □

Lemma 2.35. BoolHoist *and* LoobHoist *can be justified to be applied together as a simplification rule by clause redundancy when* σ *is the identity. Moreover, they can even be applied when their eligibility condition does not hold.*

Proof. By condition 1 of Definition 2.22, using the fact that $\models_{o\lambda}$ interprets Booleans. □

The following two rules normalize negative literals with T and $\bot$ into positive literals.

$$\frac{C' \vee s \not\approx \mathsf{T}}{C' \vee s \approx \bot}\,\textsc{NotTrue} \qquad\qquad \frac{C' \vee s \not\approx \bot}{C' \vee s \approx \mathsf{T}}\,\textsc{NotFalse}$$

Lemma 2.36. NotTrue *and* NotFalse *can be justified as simplification rules by simple clause redundancy.*

Proof. By condition 1 of Definition 2.22, using the fact that $\models_{o\lambda}$ interprets Booleans. □

The following rule is inspired by one of Leo-II's extensionality rules [8]:

$$\frac{\overbrace{C' \vee s \not\approx s'}^{C}}{C' \vee s\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,s')\ \not\approx s'\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,s')}\,\textsc{NegExt}$$

Lemma 2.37. NegExt *can be justified by simple clause redundancy.*

Proof. Let θ be a grounding substitution. We apply condition 1 of Definition 2.22. By the argument congruence property of $\models_{o\lambda}$, we have

$$\mathcal{F}\left((C' \vee s\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,s')\ \not\approx s'\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,s'))\theta\right) \models_{o\lambda} \mathcal{F}\left((C' \vee s \not\approx s')\theta\right)$$

By (O5), we have $(C' \vee s'\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,s')\ \not\approx s\ \mathsf{diff}\langle\tau,\upsilon\rangle(s,s'))\theta \prec (C' \vee s \not\approx s')\theta$. Thus condition 1 of Definition 2.22 applies. □

2.8. Examples. In this subsection, we illustrate the various rules of our calculus on concrete examples. For better readability, we use nominal λ notation.

Example 2.38 (Selection of Negated Predicates)**.** This example demonstrates the value of allowing selection of literals of the form $t \approx \bot$. Although the original λ-superposition calculus was claimed to support selection of such literals, its completeness proof was flawed in this respect [4, 17].

Consider the following clause set:

$$(1)\ \mathsf{p}\,\mathsf{a} \approx \mathsf{T}$$
$$(2)\ \mathsf{q}\,\mathsf{b} \approx \mathsf{T}$$
$$(3)\ \mathsf{r}\,\mathsf{c} \approx \mathsf{T}$$
$$(4)\ \mathsf{p}\,x \approx \bot \vee \mathsf{q}\,y \approx \bot \vee \mathsf{r}\,z \approx \bot$$

Let us first explore what happens without literal selection. Due to the variables in (4), all of the literals in (4) are incomparable w.r.t. any term order. So, since none of the literals is selected, there are three possible Sup inferences: (1) into (4), (2) into (4), and (3) into (4). After applying FalseElim to their conclusions, we obtain:

$$(5)\ \mathsf{q}\,y \approx \bot \vee \mathsf{r}\,z \approx \bot$$
$$(6)\ \mathsf{p}\,x \approx \bot \vee \mathsf{r}\,z \approx \bot$$
$$(7)\ \mathsf{p}\,x \approx \bot \vee \mathsf{q}\,y \approx \bot$$

For each of these clauses, we can again apply a Sup inference using (1), (2), or (3), in two different ways each. After applying FalseElim to their conclusions, we obtain three more clauses: $\mathsf{p}\,x \approx \bot$, $\mathsf{q}\,y \approx \bot$ and $\mathsf{r}\,z \approx \bot$. From each of these clauses, we can then derive the empty clause by another Sup and FalseElim inference. So, without literal selection, depending on the prover's heuristics, a prover might in the worst case need to perform $3 + 3 \cdot 2 + 1 = 10$ Sup inferences to derive the empty clause.

Now, let us consider the same initial clause set but we select exactly one literal whenever possible. In (4), we can select one of the literals, say the first one. Then there is only one possible Sup inference: (1) into (4), yielding (5) after applying FalseElim. In (5), we can again select the first literal. Again, only one Sup inference is possible, yielding $\mathsf{r}\,z \approx \bot$ after applying FalseElim. Another Sup and another FalseElim inference yield the empty clause. Overall, there is a unique derivation of the empty clause, consisting of only three Sup inferences.

Example 2.39 (Simplification of Functional Literals)**.** Consider the following clauses, where f and g are constants of type $\iota \to \iota$.

$$(1)\ \mathsf{f} \approx \mathsf{g}$$
$$(2)\ \mathsf{f} \not\approx \mathsf{g}$$

A Sup inference from (1) into (2) is not possible because the terms are functional. Instead, we can apply ArgCong and NegExt to derive the following clauses:

$$(3)\ \mathsf{f}\,x \approx \mathsf{g}\,x \quad \text{(by ArgCong from (1))}$$
$$(4)\ \mathsf{f}\,\mathsf{diff}(\mathsf{f},\mathsf{g}) \not\approx \mathsf{g}\,\mathsf{diff}(\mathsf{f},\mathsf{g}) \quad \text{(by NegExt from (2))}$$

Both ArgCong and NegExt are simplification rules, so we can delete (1) and (2) after deriving (3) and (4). Now, a Sup inference from (3) into (4) and a EqRes inference yield the empty clause.

In contrast, the original superposition calculus requires both the Sup inference from (1) into (2) and also a derivation similar to the one above. Moreover, its redundancy criterion does not allow us to delete (1) and (2). This amounts to doubling the number of clauses and inferences—even more if f and g had more than one argument.

Example 2.40 (Extensionality Reasoning)**.** Consider the following clauses:

(1) $\mathsf{map}\,(\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,u\,1))\,x \not\approx \mathsf{map}\,(\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,1\,u))\,x$

(2) $\mathsf{add}\,u\,v \approx \mathsf{add}\,v\,u$

For better readability, we omit type arguments and use subscripts for the parameters of diff. Using our calculus, we derive the following clauses:

(3) $\mathsf{sqrt}\,(\mathsf{add}\,(\mathsf{diff}_{\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,u\,1),z})\,1) \not\approx z\,(\mathsf{diff}_{\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,u\,1),z}) \vee$
$\mathsf{map}\,z\,x \not\approx \mathsf{map}\,(\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,1\,u))\,x$ (by Ext from (1))

(4) $\mathsf{sqrt}\,(\mathsf{add}\,\mathsf{diff}_{\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,u\,1),\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,1\,u)}\,1) \not\approx$
$\mathsf{sqrt}\,(\mathsf{add}\,1\,\mathsf{diff}_{\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,u\,1),\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,1\,u)})$ (by EqRes from (3))

(5) $\mathsf{sqrt}\,(\mathsf{add}\,1\,\mathsf{diff}_{\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,u\,1),\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,1\,u)}) \not\approx$
$\mathsf{sqrt}\,(\mathsf{add}\,1\,\mathsf{diff}_{\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,u\,1),\lambda u.\,\mathsf{sqrt}\,(\mathsf{add}\,1\,u)})$ (by Sup from (2), (4))

(6) $\bot$ (by EqRes from (5))

While such a derivation is also possible in the original λ-superposition calculus, the term orders of the original calculus were not able to compare the literals of the extensionality axiom

$$y\,\mathsf{diff}_{y,z} \not\approx z\,\mathsf{diff}_{y,z} \vee y \approx z$$

As a result, the extensionality axiom leads to an explosion of inferences. Our calculus avoids this problem by ensuring that the positive literal of the extensionality axiom is maximal, via the ordering property (O5). By replacing the extensionality axiom with the Ext rule, we avoid in addition Sup inferences into functional terms, and it strengthens our redundancy criterion.

Example 2.41 (Universal Quantification)**.** Consider the following clause set:

(1) $(\lambda x.\,\mathsf{p}\,x) \approx (\lambda x.\,\mathsf{T})$

(2) $\mathsf{p}\,\mathsf{a} \approx \bot$

Here, clause (1) encodes the universal quantification $\forall x.\,\mathsf{p}\,x$. We can derive a contradiction as follows:

(3) $p\,x \approx \mathsf{T}$ (by ArgCong from (1))

(4) $\mathsf{T} \approx \bot$ (by Sup from (2), (3))

(5) $\bot$ (by FalseElim from (4))

Since the ArgCong inference creating clause (3) can be used as a simplification rule by Lemma 2.32, clause (1) can be deleted when creating clause (3). So we do not need to apply any Ext inferences into clause (1). Except for inferences into (1) and except for a Diff inference, the inferences required in the derivation above are the only ones possible. In this sense, the encoding of the universal quantifier using λ-abstractions has no overhead.

Example 2.42 (Existential Quantification)**.** Negated universal quantification or existential quantification can be dealt with similarly. Consider the following clause set:

(1) $(\lambda x.\,\mathsf{p}\,x) \not\approx (\lambda x.\mathsf{T})$

(2) $\mathsf{p}\,x \approx \mathsf{T}$

We can derive a contradiction as follows:

(3) $\mathsf{p}\ \mathsf{diff}\langle\iota, o\rangle(\lambda x.\ \mathsf{p}\ x, \lambda x.\mathsf{T}) \not\approx \mathsf{T}$ (by NegExt from (1))
(4) $\mathsf{p}\ \mathsf{diff}\langle\iota, o\rangle(\lambda x.\ \mathsf{p}\ x, \lambda x.\mathsf{T}) \approx \bot$ (by NotTrue from (3))
(5) $\mathsf{T} \approx \bot$ (by Sup from (2), (4))
(6) $\bot$ (by FalseElim from (5))

Again, we can delete (1) when creating (3), preventing any Ext inferences from (1). Moreover, we can delete (3) when creating (4). As a result, encoding existential quantification using λ-abstraction does not have overhead either.

Example 2.43. This example illustrates why condition 2 of Sup allows u to be a variable if it has another occurrence inside of a parameter. Consider the following clause set:

(1) $\mathsf{b} \approx \mathsf{a}$
(2) $(\lambda x.\ (\neg\mathsf{p}\ x\ y) \mathbin{\mathsf{\wedge}} (\mathsf{p}\ x\ y \mathbin{\mathsf{\vee}} y \not\approx \mathsf{a})) \not\approx (\lambda x.\ \bot)$
(3) $(\lambda x.\ (\neg\mathsf{p}\ x\ \mathsf{b}) \mathbin{\mathsf{\wedge}} (\mathsf{p}\ x\ \mathsf{b} \mathbin{\mathsf{\vee}} \mathsf{b} \not\approx \mathsf{a})) \not\approx (\lambda x.\ \bot)$

Note that the clauses $(\lambda x.\ \ldots) \not\approx (\lambda x.\ \bot)$ can be read as $\exists x. \ldots$ and that (3) is an instance of (2). Clauses (1) and (3) alone are unsatisfiable because (1) ensures that the right side of the disjunction $\mathsf{p}\ x\ \mathsf{b} \mathbin{\mathsf{\vee}} \mathsf{b} \not\approx \mathsf{a}$ in (3) is false, and since $(\neg\mathsf{p}\ x\ \mathsf{b}) \mathbin{\mathsf{\wedge}} (\mathsf{p}\ x\ \mathsf{b})$ is clearly false, clause (3) is false.

For the following derivation, we assume $\mathsf{b} \succ \mathsf{a}$. Applying NegExt to (2) and (3) followed by NotFalse yields

(4) $\neg\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ y)\mathsf{\wedge}(\mathsf{p}\ x\ y\mathsf{\vee}y\not\approx\mathsf{a}),\lambda x.\ \bot}\ y \mathbin{\mathsf{\wedge}} \mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ y)\mathsf{\wedge}(\mathsf{p}\ x\ y\mathsf{\vee}y\not\approx\mathsf{a}),\lambda x.\ \bot}\ y \mathbin{\mathsf{\vee}} y \not\approx \mathsf{a} \approx \mathsf{T}$
(5) $\neg\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ \mathsf{b})\mathsf{\wedge}(\mathsf{p}\ x\ \mathsf{b}\mathsf{\vee}\mathsf{b}\not\approx\mathsf{a}),\lambda x.\ \bot}\ \mathsf{b} \mathbin{\mathsf{\wedge}} \mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ \mathsf{b})\mathsf{\wedge}(\mathsf{p}\ x\ \mathsf{b}\mathsf{\vee}\mathsf{b}\not\approx\mathsf{a}),\lambda x.\ \bot}\ \mathsf{b} \mathbin{\mathsf{\vee}} \mathsf{b} \not\approx \mathsf{a} \approx \mathsf{T}$

For better readability, we omit the type arguments and write the parameters of diff as subscripts. Applying Clausify several times yields

(6) $\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ y)\mathsf{\wedge}(\mathsf{p}\ x\ y\mathsf{\vee}y\not\approx\mathsf{a}),\lambda x.\ \bot}\ y \approx \bot$
(7) $\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ y)\mathsf{\wedge}(\mathsf{p}\ x\ y\mathsf{\vee}y\not\approx\mathsf{a}),\lambda x.\ \bot}\ y \approx \mathsf{T} \vee y \not\approx \mathsf{a}$
(8) $\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ \mathsf{b})\mathsf{\wedge}(\mathsf{p}\ x\ \mathsf{b}\mathsf{\vee}\mathsf{b}\not\approx\mathsf{a}),\lambda x.\ \bot}\ \mathsf{b} \approx \bot$
(9) $\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ \mathsf{b})\mathsf{\wedge}(\mathsf{p}\ x\ \mathsf{b}\mathsf{\vee}\mathsf{b}\not\approx\mathsf{a}),\lambda x.\ \bot}\ \mathsf{b} \approx \mathsf{T} \vee \mathsf{b} \not\approx \mathsf{a}$

By positive simplify-reflect on (9), followed by Demod from (1) into the resulting clause, we obtain the clause

(10) $\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ \mathsf{b})\mathsf{\wedge}(\mathsf{p}\ x\ \mathsf{b}\mathsf{\vee}\mathsf{b}\not\approx\mathsf{a}),\lambda x.\ \bot}\ \mathsf{a} \approx \mathsf{T}$

In this derivation, (2), (3), (4), (5), and (9) can be deleted because NegExt, NotFalse, Clausify, Demod, and positive simplify-reflect can be applied as simplification rules. Moreover, (8) can be deleted by Subsumption using (6) and a suitable relation $\sqsupset$.The following clauses remain:

(1) $\mathsf{b} \approx \mathsf{a}$
(6) $\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ y)\mathsf{\wedge}(\mathsf{p}\ x\ y\mathsf{\vee}y\not\approx\mathsf{a}),\lambda x.\ \bot}\ y \approx \bot$
(7) $\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ y)\mathsf{\wedge}(\mathsf{p}\ x\ y\mathsf{\vee}y\not\approx\mathsf{a}),\lambda x.\ \bot}\ y \approx \mathsf{T} \vee y \not\approx \mathsf{a}$
(10) $\mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg\mathsf{p}\ x\ \mathsf{b})\mathsf{\wedge}(\mathsf{p}\ x\ \mathsf{b}\mathsf{\vee}\mathsf{b}\not\approx\mathsf{a}),\lambda x.\ \bot}\ \mathsf{a} \approx \mathsf{T}$

Assuming that the negative literal in (7) is selected and that $\mathsf{b} \succ \mathsf{a}$, we now need a Sup inference from (1) into the variable y in the second literal of (7), which is possible because y also appears in a parameter in (7). This Sup inference yields:

$$(11)\ \mathsf{p}\ \mathsf{diff}_{\lambda x.\ (\neg \mathsf{p}\ x\ \mathsf{b}) \wedge (\mathsf{p}\ x\ \mathsf{b} \vee \mathsf{b} \not\approx \mathsf{a}), \lambda x.\ \bot}\ \mathsf{b} \approx \mathsf{T} \vee \mathsf{a} \not\approx \mathsf{a}$$

The empty clause can then be derived using EqRes, a Sup inference with (6), and FalseElim.

3. Soundness

To prove our calculus sound, we need a substitution lemma for terms and clauses, which our logic fulfills:

Lemma 3.1 (Substitution Lemma)**.** *Let θ be a substitution, and let t be a term of type τ. For any proper interpretation $\mathcal{I} = (\mathcal{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$ and any valuation ξ,*

$$[\![t\theta]\!]^{\xi}_{\mathcal{J}} = [\![t]\!]^{\xi'}_{\mathcal{J}}$$

where the modified valuation ξ' is defined by $\xi'_{\mathsf{ty}}(\alpha) = [\![\alpha\theta]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}$ for type variables α and $\xi'_{\mathsf{te}}(x) = [\![x\theta]\!]^{\xi}_{\mathcal{J}}$ for term variables x.

Proof. By induction on the size of the term t.

Case $t = x\langle\tau\rangle$:

$$\begin{aligned}
[\![t\theta]\!]^{\xi}_{\mathcal{J}} &= [\![x\theta]\!]^{\xi}_{\mathcal{J}} \\
&= \xi'(x) \quad \text{(by the definition of interpretation)} \\
&= [\![x]\!]^{\xi'}_{\mathcal{J}} \quad \text{(since } x \text{ is mapped to } [\![x\theta]\!]^{\xi}_{\mathcal{J}}\text{)} \\
&= [\![t]\!]^{\xi'}_{\mathcal{J}}
\end{aligned}$$

Case $t = \mathsf{f}\langle\bar{\tau}\rangle(\bar{u})$:

$$\begin{aligned}
[\![t\theta]\!]^{\xi}_{\mathcal{J}} &= [\![\mathsf{f}\langle\bar{\tau}\theta\rangle(\bar{u}\theta)]\!]^{\xi}_{\mathcal{J}} \\
&= \mathcal{J}\,(\mathsf{f}, [\![\bar{\tau}\theta]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}, [\![\bar{u}\theta]\!]^{\xi}_{\mathcal{J}}) \quad \text{(by definition)} \\
&= \mathcal{J}\,(\mathsf{f}, [\![\bar{\tau}]\!]^{\xi'_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}, [\![\bar{u}]\!]^{\xi'}_{\mathcal{J}}) \quad \text{(by induction hypothesis)} \\
&= [\![\mathsf{f}\langle\bar{\tau}\rangle(\bar{u})]\!]^{\xi'}_{\mathcal{J}} \quad \text{(by definition)} \\
&= [\![t]\!]^{\xi'}_{\mathcal{J}}
\end{aligned}$$

Case $t = s\ v$:

$$\begin{aligned}
[\![t\theta]\!]^{\xi}_{\mathcal{J}} &= [\![s\theta\ v\theta]\!]^{\xi}_{\mathcal{J}} \\
&= [\![s\theta]\!]^{\xi}_{\mathcal{J}}\,([\![v\theta]\!]^{\xi}_{\mathcal{J}}) \quad \text{(by definition)} \\
&= [\![s]\!]^{\xi'}_{\mathcal{J}}\,([\![v]\!]^{\xi'}_{\mathcal{J}}) \quad \text{(by induction hypothesis)} \\
&= [\![s\ v]\!]^{\xi'}_{\mathcal{J}} \quad \text{(by definition)} \\
&= [\![t]\!]^{\xi'}_{\mathcal{J}}
\end{aligned}$$

CASE $t = \lambda\langle\tau\rangle\ u$:

$$\begin{aligned}
[\![t\theta]\!]^{\xi}_{\mathcal{I}}(a) &= [\![\lambda\langle\tau\theta\rangle\ u\theta]\!]^{\xi}_{\mathcal{I}}(a) \\
&= [\![u\theta\{0 \mapsto x\}]\!]^{(\xi_{\mathsf{ty}},\xi_{\mathsf{te}}[x\mapsto a])}_{\mathcal{I}} \quad \text{(since } \mathcal{I} \text{ is proper; for some fresh variable } x\text{)} \\
&= [\![u\{0 \mapsto x\}\theta]\!]^{(\xi_{\mathsf{ty}},\xi_{\mathsf{te}}[x\mapsto a])}_{\mathcal{I}} \\
&= [\![u\{0 \mapsto x\}]\!]^{(\xi'_{\mathsf{ty}},\xi'_{\mathsf{te}}[x\mapsto a])}_{\mathcal{I}} \quad \text{(by induction hypothesis)} \\
&= [\![\lambda\langle\tau\rangle\ u]\!]^{\xi'}_{\mathcal{I}}(a) \quad \text{(since } \mathcal{I} \text{ is proper)} \\
&= [\![t]\!]^{\xi'}_{\mathcal{I}}(a)
\end{aligned}$$

□

Lemma 3.2 (Substitution Lemma for Clauses)**.** *Let θ be a substitution, and let C be a clause. For any proper interpretation $\mathcal{I} = (\mathcal{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$ and any valuation ξ, $C\theta$ is true w.r.t. $\mathcal{I}$ and ξ if and only if C is true w.r.t. $\mathcal{I}$ and ξ', where the modified valuation ξ' is defined by $\xi'_{\mathsf{ty}}(\alpha) = [\![\alpha\theta]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}$ for type variables α and $\xi'_{\mathsf{te}}(x) = [\![x\theta]\!]^{\xi}_{\mathcal{I}}$ for term variables x.*

Proof. By definition of the semantics of clauses, $C\theta$ is true w.r.t. $\mathcal{I}$ and ξ if and only if one of its literals is true w.r.t. $\mathcal{I}$ and ξ. By definition of the semantics of literals, a positive literal $s\theta \approx t\theta$ (resp. negative literal $s\theta \not\approx t\theta$) of $C\theta$ is true w.r.t. $\mathcal{I}$ and ξ if and only if $[\![s\theta]\!]^{\xi}_{\mathcal{I}}$ and $[\![t\theta]\!]^{\xi}_{\mathcal{I}}$ are equal (resp. different). By Lemma 3.1, $[\![s\theta]\!]^{\xi}_{\mathcal{I}}$ and $[\![t\theta]\!]^{\xi}_{\mathcal{I}}$ are equal (resp. different) if and only if $[\![s]\!]^{\xi'}_{\mathcal{I}}$ and $[\![t]\!]^{\xi'}_{\mathcal{I}}$ are equal (resp. different)—i.e., if and only if a literal $s \approx t$ (resp. $s \not\approx t$) in C is true w.r.t. $\mathcal{I}$ and ξ'. This holds if and only if C is true w.r.t. $\mathcal{I}$ and ξ'. □

Theorem 3.3. *All core inference rules are sound w.r.t. $\approx$ (Definition 1.1). All core inference rules except for* EXT*,* FLUIDEXT*, and* DIFF *are also sound w.r.t. $\models$. This holds even when ignoring order, selection, and eligibility conditions.*

Proof. We fix an inference and an interpretation $\mathcal{I}$ that is a model of the premises. For EXT, FLUIDEXT, and DIFF inferences, we assume that $\mathcal{I}$ is diff-aware. We need to show that it is also a model of the conclusion. By Lemma 3.2, $\mathcal{I}$ is a model of the σ-instances of the premises as well, where σ is the substitution used for the inference. From the semantics of our logic, it is easy to see that congruence holds at green positions and at the left subterm of an application. To show that $\mathcal{I}$ is a model of the conclusion, it suffices to show that the conclusion is true under $\mathcal{I}, \xi$ for all valuations ξ.

For most rules, it suffices to make distinctions on the truth under $\mathcal{I}, \xi$ of the literals of the σ-instances of the premises, to consider the conditions that σ is a unifier where applicable, and to apply congruence. For BOOLHOIST, LOOBHOIST, FALSEELIM, CLAUSIFY, FLUIDBOOLHOIST, FLUIDLOOBHOIST, we also use the fact that $\mathcal{I}$ interprets logical symbols correctly. For EXT, FLUIDEXT, and DIFF, we also use the assumption that $\mathcal{I}$ is diff-aware. □

4. Refutational Completeness

4.1. Logics and Encodings. In our completeness proof, we use two higher-order signatures and one first-order signature.

Let Σ_{H} be the higher-order signature used by the calculus described in Section 2. It is required to contain a symbol $\mathsf{diff} : \Pi\alpha, \beta.\ (\alpha \to \beta, \alpha \to \beta) \Rightarrow \alpha$

Let Σ_{I} be the signature obtained from Σ_{H} in the following way: We replace each constant with parameters $\mathsf{f} : \Pi\bar{\alpha}_m.\ \bar{\tau}_n \Rightarrow \tau$ in Σ_{H} with a family of constants $\mathsf{f}^{\bar{\upsilon}_m}_{\bar{t}_n} : \tau$, indexed

by all possible ground types $\bar{\upsilon}_m$ and ground terms $\bar{t}_n \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$ of type $\bar{\tau}_n\{\bar{\alpha}_m \mapsto \bar{\upsilon}_m\}$. Constants without parameters (even those with type arguments) are left as they are.

In some contexts, it is more convenient to use terms from $\mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$ instead of $\mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$ in the subscripts t_i of the constants $\mathsf{f}_{\bar{t}_n}^{\bar{\upsilon}_m}$. We follow this convention:

Convention 4.1. In the subscripts t_i of constants $\mathsf{f}_{\bar{t}_n}^{\bar{\upsilon}_m} \in \Sigma_{\text{I}}$, we identify each term of the form $\mathsf{f}\langle\bar{\upsilon}_m\rangle(\bar{t}_n) \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$ with the term $\mathsf{f}_{\bar{t}_n}^{\bar{\upsilon}_m} \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$, whenever $n > 0$.

Similarly, the first-order signatures $\mathcal{F}(\Sigma_{\text{I}})$ and $\mathcal{F}(\Sigma_{\text{H}})$ as defined in Section 2.6 are almost identical, the only difference being that the subscripts t of the symbols $\mathsf{fun}_t \in \mathcal{F}(\Sigma_{\text{H}})$ may contain symbols with parameters, whereas the subscripts t of the symbols $\mathsf{fun}_t \in \mathcal{F}(\Sigma_{\text{I}})$ may not. To repair this mismatch, we adopt the following convention using the obvious correspondence between the symbols in $\mathcal{F}(\Sigma_{\text{H}})$ and $\mathcal{F}(\Sigma_{\text{I}})$:

Convention 4.2. In the subscripts of constants fun_t in $\mathcal{F}(\Sigma_{\text{H}})$ and $\mathcal{F}(\Sigma_{\text{I}})$, we identify each term of the form $\mathsf{f}\langle\bar{\upsilon}_m\rangle(\bar{t}_n) \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$ with the term $\mathsf{f}_{\bar{t}_n}^{\bar{\upsilon}_m} \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$, whenever $n > 0$. Using this identification, we can consider the first-order signatures $\mathcal{F}(\Sigma_{\text{H}})$ and $\mathcal{F}(\Sigma_{\text{I}})$ to be identical.

The table below summarizes our completeness proof's four levels, each with a set of terms and a set of clauses. We write $\mathcal{T}_{\text{X}}$ for the set of terms and $\mathcal{C}_{\text{X}}$ for the set of clauses of a given level X:

Level	Terms	Clauses
F	ground first-order terms over $\mathcal{F}(\Sigma_{\text{I}})$	clauses over $\mathcal{T}_{\text{F}}$
IG	$\mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$	clauses over $\mathcal{T}_{\text{IG}}$
G	$\mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$	clauses over $\mathcal{T}_{\text{G}}$
H	$\mathcal{T}(\Sigma_{\text{H}})$	clauses over $\mathcal{T}_{\text{H}}$

4.1.1. *First-Order Encoding.* We use the map $\mathcal{F}$ defined in Definition 2.19 both as an encoding from $\mathcal{T}_{\text{IG}}/\mathcal{C}_{\text{IG}}$ to $\mathcal{T}_{\text{F}}/\mathcal{C}_{\text{F}}$ and as an encoding from $\mathcal{T}_{\text{G}}/\mathcal{C}_{\text{G}}$ to $\mathcal{T}_{\text{F}}/\mathcal{C}_{\text{F}}$. Potential for confusion is minimal because the two encodings coincide on the values that are in the domain of both.

Lemma 4.3. *A term $s \in \mathcal{T}_{\text{IG}}$ is a green subterm of $t \in \mathcal{T}_{\text{IG}}$ if and only if $\mathcal{F}(s)$ is a subterm of $\mathcal{F}(t)$.*

Proof. By induction using the definition of $\mathcal{F}$. □

4.1.2. *Indexing of Parameters.*

Definition 4.4 (Indexing of Parameters)**.** The transformation $\mathcal{J}$ translates from $\mathcal{T}_{\text{G}}$ to $\mathcal{T}_{\text{IG}}$ by encoding any occurrence of a constant with parameters $\mathsf{f}\langle\bar{\upsilon}\rangle(\bar{u})$ as $\mathsf{f}_{\bar{u}}^{\bar{\upsilon}}$. Formally:

$$\begin{aligned}
\mathcal{J}(x) &= x\\
\mathcal{J}(\lambda\, t) &= \lambda\, \mathcal{J}(t)\\
\mathcal{J}(\mathsf{f}\langle\bar{\upsilon}\rangle\ \bar{s}) &= \mathsf{f}\langle\bar{\upsilon}\rangle\ \mathcal{J}(\bar{s})\\
\mathcal{J}(\mathsf{f}\langle\bar{\upsilon}\rangle\ (\bar{u}_k)\ \bar{s}) &= \mathsf{f}_{\bar{u}_k}^{\bar{\upsilon}}\ \mathcal{J}(\bar{s}) \text{ if } k > 0\\
\mathcal{J}(m\ \bar{s}) &= m\ \mathcal{J}(\bar{s})
\end{aligned}$$

We extend $\mathcal{J}$ to clauses by mapping each side of each literal individually.

Lemma 4.5. *Let $t \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$, and let C be a clause over $\mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$. Then $\mathcal{F}(\mathcal{J}(t)) = \mathcal{F}(t)$ and $\mathcal{F}(\mathcal{J}(C)) = \mathcal{F}(C)$.*

Proof. For t, the claim follows directly from the definitions of $\mathcal{J}$ (Definition 4.4) and $\mathcal{F}$ (Definition 2.19), relying on the identification of fun_t and $\mathsf{fun}_{\mathcal{J}(t)}$ (Convention 4.2). For C, the claim holds because $\mathcal{J}$ and $\mathcal{F}$ map each side of each literal individually. □

4.2. Calculi. In this section, we define the calculi *IGInf* and *GInf*, for the respective levels IG and G. Both of these calculi are parameterized by a relation $\succ$ on ground terms and ground clauses and by a selection function *sel*. The specific requirements on $\succ$ depend on the calculus and are given in the corresponding subsection below.

For the F level, we use the calculus $\mathit{FInf}^{\succ,\mathit{sel}}$ introduced in Section 2.6. We require that $\succ$ is an admissible term order for *FInf* in the following sense:

Definition 4.6. Let $\succ$ be a relation on ground terms and ground clauses. Such a relation $\succ$ is an *admissible term order for FInf* if it fulfills the following properties:

- $(\text{O1})_{\text{F}}$ the relation $\succ$ on ground terms is a well-founded total order;
- $(\text{O2})_{\text{F}}$ ground compatibility with contexts: if $s' \succ s$, then $s'[t] \succ s[t]$;
- $(\text{O3})_{\text{F}}$ ground subterm property: $t[s] \succ s$ for ground terms s and t;
- $(\text{O4})_{\text{F}}$ $u \succ \bot \succ \top$ for all ground terms $u \notin \{\top, \bot\}$;
- $(\text{O5})_{\text{F}}$ $\mathcal{F}(u) \succ \mathcal{F}(u\,\mathsf{diff}^{\tau,\upsilon}_{s,t})$ for all $s, t, u : \tau \to \upsilon \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$;
- $(\text{O6})_{\text{F}}$ the relation $\succ$ on ground clauses is the standard extension of $\succ$ on ground terms via multisets [1, Sect. 2.4];

Remark 4.7. By Lemma 4.5 and because $\mathcal{J}$ is a bijection, the rules FArgCong, FExt, and FDiff, can equivalently be described by using s, s', u, w from $\mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$ instead of $\mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$ and replacing $\mathsf{diff}\langle\tau,\upsilon\rangle(u,w)$ with $\mathsf{diff}^{\tau,\upsilon}_{u,w}$.

4.2.1. *Indexed Ground Higher-Order Level.* The calculus $\mathit{IGInf}^{\succ,\mathit{sel}}$ is parameterized by a relation $\succ$ and a selection function *sel*. We require that $\succ$ is an admissible term order for *IGInf* in the following sense:

Definition 4.8. Let $\succ$ be a relation on $\mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$, and on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$. Such a relation $\succ$ is *an admissible term order for IGInf* if it fulfills the following properties:

- $(\text{O1})_{\text{IG}}$ the relation $\succ$ on ground terms is a well-founded total order;
- $(\text{O2})_{\text{IG}}$ ground compatibility with yellow contexts: $s' \succ s$ implies $t\langle\!\langle s' \rangle\!\rangle \succ t\langle\!\langle s \rangle\!\rangle$ for ground terms s, s', and t;
- $(\text{O3})_{\text{IG}}$ ground yellow subterm property: $t\langle\!\langle s \rangle\!\rangle \succeq s$ for ground terms s and t;
- $(\text{O4})_{\text{IG}}$ $u \succ \bot \succ \top$ for all ground terms $u \notin \{\top, \bot\}$;
- $(\text{O5})_{\text{IG}}$ $u \succ u\,\mathsf{diff}^{\tau,\upsilon}_{s,t}$ for all ground terms $s, t, u : \tau \to \upsilon$.
- $(\text{O6})_{\text{IG}}$ the relation $\succ$ on ground clauses is the standard extension of $\succ$ on ground terms via multisets [1, Sect. 2.4];

The rules of $\mathit{IGInf}^{\succ,\mathit{sel}}$ (abbreviated *IGInf*) are the following.

$$\frac{\overbrace{D' \vee t \approx t'}^{D} \quad C\langle t \rangle}{D' \vee C\langle t' \rangle}\,\text{IGSup}$$

with the following side conditions:

1. t is nonfunctional;
2. $t \succ t'$;
3. $D \prec C\langle t \rangle$;
4. the position of t is eligible in C;
5. $t \approx t'$ is strictly eligible in D;
6. if t' is Boolean, then $t' = \mathsf{T}$.

$$\frac{\overbrace{C' \vee u \not\approx u}^{C}}{C'}\,\text{IGEqRes} \qquad \frac{\overbrace{C' \vee u \approx v' \vee u \approx v}^{C}}{C' \vee v \not\approx v' \vee u \approx v'}\,\text{IGEqFact}$$

Side conditions for IGEqRes:

1. $u \not\approx u$ is eligible in C.

Side conditions for IGEqFact:

1. $u \approx v$ is maximal in C;
2. there are no selected literals in C;
3. $u \succ v$.

$$\frac{C' \vee s \approx t}{C' \vee D}\,\text{IGClausify}$$

with the following side conditions:

1. $s \approx t$ is strictly eligible in $C' \vee s \approx t$;
2. The triple (s, t, D) has one of the following forms, where τ is an arbitrary type and u, v are arbitrary terms:

$(u \mathbin{\mathsf{\wedge}} v,\ \mathsf{T},\ u \approx \mathsf{T})$	$(u \mathbin{\mathsf{\wedge}} v,\ \mathsf{T},\ v \approx \mathsf{T})$	$(u \mathbin{\mathsf{\wedge}} v,\ \bot,\ u \approx \bot \vee v \approx \bot)$
$(u \mathbin{\mathsf{\vee}} v,\ \mathsf{T},\ u \approx \mathsf{T} \vee v \approx \mathsf{T})$	$(u \mathbin{\mathsf{\vee}} v,\ \bot,\ u \approx \bot)$	$(u \mathbin{\mathsf{\vee}} v,\ \bot,\ v \approx \bot)$
$(u \mathbin{\mathsf{\rightarrow}} v,\ \mathsf{T},\ u \approx \bot \vee v \approx \mathsf{T})$	$(u \mathbin{\mathsf{\rightarrow}} v,\ \bot,\ u \approx \mathsf{T})$	$(u \mathbin{\mathsf{\rightarrow}} v,\ \bot,\ v \approx \bot)$
$(u \mathbin{\mathsf{\approx}^\tau} v,\ \mathsf{T},\ u \approx v)$	$(u \mathbin{\mathsf{\approx}^\tau} v,\ \bot,\ u \not\approx v)$	
$(u \mathbin{\mathsf{\not\approx}^\tau} v,\ \mathsf{T},\ u \not\approx v)$	$(u \mathbin{\mathsf{\not\approx}^\tau} v,\ \bot,\ u \approx v)$	
$(\mathsf{\neg} u,\ \mathsf{T},\ u \approx \bot)$	$(\mathsf{\neg} u,\ \bot,\ u \approx \mathsf{T})$	

$$\frac{C\langle u \rangle}{C\langle \bot \rangle \vee u \approx \mathsf{T}}\,\text{IGBoolHoist} \qquad \frac{C\langle u \rangle}{C\langle \mathsf{T} \rangle \vee u \approx \bot}\,\text{IGLoobHoist}$$

each with the following side conditions:

1. u is of Boolean type;
2. u is neither T nor $\bot$;
3. the position of u is eligible in C;
4. the occurrence of u is not in a literal of the form $u \approx \bot$ or $u \approx \mathsf{T}$.

$$\frac{\overbrace{C' \vee \bot \approx \top}^{C}}{C'}\,\text{IGFalseElim}$$

with the following side conditions:
1. $\bot \approx \top$ is strictly eligible in C.

$$\frac{\overbrace{C' \vee s \approx s'}^{C}}{C' \vee s\,\mathsf{diff}^{\tau,\upsilon}_{u,w} \approx s'\,\mathsf{diff}^{\tau,\upsilon}_{u,w}}\,\text{IGArgCong}$$

with the following side conditions:
1. s is of type $\tau \to \upsilon$;
2. u, w are ground terms of type $\tau \to \upsilon$;
3. $s \approx s'$ is strictly eligible in C.

$$\frac{C\langle u\rangle}{C\langle w\rangle \vee u\,\mathsf{diff}^{\tau,\upsilon}_{u,w} \not\approx w\,\mathsf{diff}^{\tau,\upsilon}_{u,w}}\,\text{IGExt}$$

with the following side conditions:
1. the position of u is eligible in $C\langle u\rangle$;
2. the type of u is $\tau \to \upsilon$;
3. w is a ground term of type $\tau \to \upsilon$;
4. $u \succ w$;

$$\frac{}{u\,\mathsf{diff}^{\tau,\upsilon}_{u,w} \not\approx u\,\mathsf{diff}^{\tau,\upsilon}_{u,w} \vee u\,s \approx w\,s}\,\text{IGDiff}$$

with the following side conditions:
1. τ and υ are ground types;
2. u, w are ground terms of type $\tau \to \upsilon$;
3. s is a ground term of type τ.

4.2.2. *Ground Higher-Order Level.* Like on the other levels, the calculus *GInf* is parameterized by a relation $\succ$ and a selection function *sel*.

Definition 4.9. Let $\succ$ be a relation on $\mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$and on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_{\text{H}})$. Such a relation $\succ$ is *an admissible term order for GInf* if it fulfills the following properties:

- $(\text{O1})_{\text{G}}$ the relation $\succ$ on ground terms is a well-founded total order;
- $(\text{O2})_{\text{G}}$ ground compatibility with yellow contexts: $s' \succ s$ implies $t\langle\!\langle s'\rangle\!\rangle \succ t\langle\!\langle s\rangle\!\rangle$ for ground terms s, s', and t;
- $(\text{O3})_{\text{G}}$ ground yellow subterm property: $t\langle\!\langle s\rangle\!\rangle \succeq s$ for ground terms s and t;
- $(\text{O4})_{\text{G}}$ $u \succ \bot \succ \top$ for all ground terms $u \notin \{\top, \bot\}$;
- $(\text{O5})_{\text{G}}$ $u \succ u\,\mathsf{diff}\langle\tau, \upsilon\rangle(s, t)$ for all ground terms $s, t, u : \tau \to \upsilon$.
- $(\text{O6})_{\text{G}}$ the relation $\succ$ on ground clauses is the standard extension of $\succ$ on ground terms via multisets [1, Sect. 2.4];

The calculus rules of *GInf* are a verbatim copy of those of *IGInf*, with the following exceptions:

– *GInf* uses Σ_{H} instead of Σ_{I} and $\mathcal{C}_{\mathrm{G}}$ instead of $\mathcal{C}_{\mathrm{IG}}$.
– The rules are prefixed by G instead of IG.
– GArgCong uses $\mathsf{diff}\langle\tau,\upsilon\rangle(u,w)$ instead of $\mathsf{diff}^{\tau,\upsilon}_{u,w}$.
– GExt uses $\mathsf{diff}\langle\tau,\upsilon\rangle(u,w)$ instead of $\mathsf{diff}^{\tau,\upsilon}_{u,w}$.
– GDiff uses $\mathsf{diff}\langle\tau,\upsilon\rangle(u,w)$ instead of $\mathsf{diff}^{\tau,\upsilon}_{u,w}$.

4.3. Redundancy Criteria and Saturation. In this subsection, we define redundancy criteria for the levels F, IG, and G and show that saturation up to redundancy on one level implies saturation up to redundancy on the previous level. We will use these results in Section 4.4 to lift refutational completeness from level F to level H.

Definition 4.10. A set N of clauses is called *saturated up to redundancy* if every inference with premises in N is redundant w.r.t. N.

4.3.1. *First-Order Level.* In this subsection, let $\succ$ be an admissible term order for *FInf* (Definition 4.6), and let *fsel* be a selection function on $\mathcal{C}_{\mathrm{F}}$.

Definition 4.11 (Inference Redundancy)**.** Given $\iota \in \mathit{FInf}$ and $N \subseteq \mathcal{C}_{\mathrm{F}}$, let $\iota \in \mathit{FRed}_{\mathrm{I}}(N)$ if ι is a Diff inference and $N \models_{\mathrm{o}\lambda} \mathit{concl}(\iota)$ or if ι is not a Diff inference and $\{E \in N \mid E \prec \mathit{mprem}(\iota)\} \models_{\mathrm{o}\lambda} \mathit{concl}(\iota)$.

4.3.2. *Indexed Ground Higher-Order Level.* In this subsubsection, let $\succ$ be an admissible term order for *IGInf* (Definition 4.8), and let *igsel* be a selection function on $\mathcal{C}_{\mathrm{IG}}$.

To lift the notion of inference redundancy, we need to connect the inference systems *FInf* and *IGInf* as follows.

Lemma 4.12. *Since $\succ$ is an admissible term order for IGInf (Definition 4.8), the relation $\succ_{\mathcal{F}}$ defined in Definition 2.21 is an admissible term order for FInf (Definition 4.6).*

Proof. This is easy to see, considering that $\mathcal{F}$ is a bijection between $\mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$ and $\mathcal{T}_{\mathrm{F}}$ (Lemma 2.20), that every first-order subterm corresponds to a higher-order yellow subterm by Lemma 4.3, and that $\mathcal{F}$ maps each side of each literal individually. □

Definition 4.13. We extend $\mathcal{F}$ to inference rules by mapping an inference $\iota \in \mathit{IGInf}$ to the inference

$$\frac{\mathcal{F}(\mathit{prems}(\iota))}{\mathcal{F}(\mathit{concl}(\iota))}$$

Lemma 4.14. *The mapping $\mathcal{F}$ is a bijection between $\mathit{IGInf}^{\succ,\mathit{igsel}}$ and $\mathit{FInf}^{\succ_{\mathcal{F}},\mathcal{F}(\mathit{igsel})}$, where $\mathcal{F}(\mathit{igsel})$ is defined in Definition 2.23.*

Proof. This is easy to see by comparing the rules of *IGInf* and *FInf* and considering Remark 4.7. It is crucial that the following concepts match:

– Subterms on the F level correspond to green subterms on the IG level by Lemma 4.3.
– The term orders correspond (Definition 2.21).

– The selected literals correspond; i.e., a literal L is selected in a clause C if and only if the literal $\mathcal{F}(L)$ is selected in $\mathcal{F}(C)$. This follows directly from the definition of $\mathcal{F}(\mathit{igsel})$ (Definition 2.23).
– The concepts of eligibility correspond; i.e., a literal L of a clause $C \in \mathcal{C}_{\mathrm{IG}}$ is (strictly) eligible w.r.t. $\succeq$ if and only if the literal $\mathcal{F}(L)$ of the clause C is (strictly) eligible w.r.t. $\succeq_{\mathcal{F}}$; and a position $L.s.p$ of a clause $C \in \mathcal{C}_{\mathrm{IG}}$ is eligible w.r.t. $\succeq$ if and only if the position $\mathcal{F}(L).\mathcal{F}(s).p$ of the clause $\mathcal{F}(C)$ is eligible w.r.t. $\succeq_{\mathcal{F}}$. This is true because eligibility (Definition 2.15) depends only on the selected literals and the term order, which correspond as discussed above. □

Definition 4.15 (Inference Redundancy)**.** Given $\iota \in \mathit{IGInf}^{\succ,\mathit{igsel}}$ and $N \subseteq \mathcal{C}_{\mathrm{IG}}$, let $\iota \in \mathit{IGRed}_{\mathrm{I}}(N)$ if $\mathcal{F}(\iota) \in \mathit{FRed}_{\mathrm{I}}(\mathcal{F}(N))$ (Definition 4.11) w.r.t. $\succ_{\mathcal{F}}$.

Using the bijection between *IGInf* and *FInf*, we can show that saturation w.r.t. *IGInf* implies saturation w.r.t. *FInf*:

Lemma 4.16. *Let N be saturated up to redundancy w.r.t. $\mathit{IGInf}^{\succ,\mathit{igsel}}$. Then $\mathcal{F}(N)$ is saturated up to redundancy w.r.t. $\mathit{FInf}^{\succ_{\mathcal{I}},\mathcal{F}(\mathit{igsel})}$.*

Proof. By Lemma 4.14 and Definition 4.15. □

4.3.3. *Ground Higher-Order Level.* In this subsubsection, let $\succ$ be an admissible term order for *GInf* (Definition 4.9), and let *gsel* be a selection function on $\mathcal{C}_{\mathrm{G}}$.

Since mapping $\mathcal{I}$ is clearly bijective, we can transfer $\succ$ from the G level to the IG level as follows:

Definition 4.17. Let $\succ$ be a relation on $\mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{H}})$ and on clauses over $\mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{H}})$. We define a relation $\succ_{\mathcal{I}}$ on $\mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$ and on clauses over $\mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$ as $d \succ_{\mathcal{I}} e$ if and only if $\mathcal{I}^{-1}(d) \succ \mathcal{I}^{-1}(e)$ for all terms or clauses d and e.

Lemma 4.18. *Since $\succ$ is an admissible term order for GInf (Definition 4.9), the relation $\succ_{\mathcal{I}}$ is an admissible term order for IGInf (Definition 4.8).*

Proof. This is easy to see, considering that $\mathcal{I}$ is a bijection and that $\mathcal{I}$ and $\mathcal{I}^{-1}$ preserve yellow subterms. □

Since $\mathcal{I}$ is bijective, we can transfer the selection function as follows:

Definition 4.19. Based on *gsel*, we define $\mathcal{I}(\mathit{gsel})$ as a selection function that selects the literals of $C \in \mathcal{C}_{\mathrm{IG}}$ corresponding to the *gsel*-selected literals in $\mathcal{I}^{-1}(C)$.

Definition 4.20. We extend $\mathcal{I}$ to inference rules by mapping an inference $\iota \in \mathit{GInf}$ to the inference

$$\frac{\mathcal{I}(\mathit{prems}(\iota))}{\mathcal{I}(\mathit{concl}(\iota))}$$

Lemma 4.21. *The mapping $\mathcal{I}$ is a bijection between $\mathit{GInf}^{\succ,\mathit{gsel}}$ and $\mathit{IGInf}^{\succ_{\mathcal{I}},\mathcal{I}(\mathit{gsel})}$.*

Proof. This is easy to see by comparing the rules of *GInf* and *IGInf*. It is crucial that the following concepts match:

– Green subterms on the IG level correspond to green subterms on the G level.
– The term orders correspond (Definition 4.17).

– The selected literals correspond; i.e., a literal L is selected in a clause C if and only if the literal $\mathcal{J}(L)$ is selected in $\mathcal{J}(C)$. This follows directly from the definition of $\mathcal{J}(gsel)$ (Definition 4.19).
– The concepts of eligibility correspond; i.e., a literal L of a clause $C \in \mathcal{C}_{\mathrm{G}}$ is (strictly) eligible w.r.t. $\succeq$ if and only if the literal $\mathcal{J}(L)$ of the clause C is (strictly) eligible w.r.t. $\succeq_{\mathcal{J}}$; and a position $L.s.p$ of a clause $C \in \mathcal{C}_{\mathrm{G}}$ is eligible w.r.t. $\succeq$ if and only if the position $\mathcal{J}(L).\mathcal{J}(s).p$ of the clause $\mathcal{J}(C)$ is eligible w.r.t. $\succeq_{\mathcal{J}}$. This is true because eligibility (Definition 2.15) depends only on the selected literals and the term order, which correspond as discussed above. □

Definition 4.22 (Inference Redundancy)**.** Let $N \subseteq \mathcal{C}_{\mathrm{G}}$ and $\iota \in \mathit{GInf}$. We define $\iota \in \mathit{GRed}_{\mathrm{I}}$ if $\mathcal{J}(\iota) \in \mathit{IGRed}_{\mathrm{I}}(\mathcal{J}(N))$.

Lemma 4.23. *Let $N \subseteq \mathcal{C}_{\mathrm{G}}$ be saturated up to redundancy w.r.t. $\mathit{GInf}^{\succ,gsel}$. Then $\mathcal{J}(N)$ is saturated up to redundancy w.r.t. $\mathit{IGInf}^{\succ_{\mathcal{J}},\mathcal{J}(gsel)}$.*

Proof. By Lemma 4.21 and Defintion 4.22. □

4.3.4. *Full Higher-Order Level.* In this subsubsection, let $\succ$ be an admissible term order (Definition 2.12) and let *hsel* be a selection function (Definition 2.14).

Lemma 4.24. *The relation $\succ$ is an admissible term order for GInf.*

Proof. Conditions (O1) to (O6) are identical to conditions $(\mathrm{O1})_{\mathrm{G}}$ to $(\mathrm{O6})_{\mathrm{G}}$. □

The selection function is transferred as follows:

Definition 4.25. Let $N \subseteq \mathcal{C}_{\mathrm{H}}$. We choose a function $\mathcal{G}_N^{-1}$, depending on this set N, such that $\mathcal{G}_N^{-1}(C) \in N$ and $\mathcal{G}(\mathcal{G}_N^{-1}(C)) = C$ for all $C \in \mathcal{G}(N)$. Then we define $\mathcal{G}(\mathit{hsel}, N)$ as a selection function that selects the literals of $C \in \mathcal{G}(N)$ corresponding to the *hsel*-selected literals in $\mathcal{G}_N^{-1}(C)$ and that selects arbitrary literals in all other clauses.

Lemma 4.26 (Lifting of Order Conditions)**.** *Let $t, s \in \mathcal{T}(\Sigma_{\mathrm{H}})$, and let ζ be a grounding substitution. If $t\zeta \succ s\zeta$, then $t \not\preceq s$. The same holds for literals.*

Proof. We prove the contrapositive. If $t \preceq s$, then, by (O7), $t\zeta \preceq s\zeta$. Therefore, since $\succ$ is asymmetric by (O1), $t\zeta \not\succ s\zeta$. The proof for literals is analogous, using (O6) and (O8). □

Lemma 4.27 (Lifting of Maximality Conditions)**.** *Let $C \in \mathcal{C}_{\mathrm{H}}$. Let θ be a grounding substitution. Let L_0 be (strictly) maximal in $C\theta$. Then there exists a literal L that is (strictly) maximal in C such that $L\theta = L_0$.*

Proof. By Definition 2.13, a literal L of a clause C is maximal if for all $K \in C$ such that $K \succeq L$, we have $K \preceq L$.

Since $L_0 \in C\theta$, there exist literals L in C such that $L\theta = L_0$. Let L be a maximal one among these literals. A maximal one must exist because $\succ$ is transitive on literals by (O9) and transitivity implies existence of maximal elements in nonempty finite sets. Let K be a literal in C such that $K \succeq L$. We must show that $K \preceq L$. By Lemma 4.26, $K\theta \not\succ L\theta = L_0$. By (O1), $\succ$ is a total order on ground terms, and thus $K\theta \succeq L_0$. By maximality of L_0 in $C\theta$, we have $K\theta \preceq L_0$ and thus $K\theta = L_0$ by (O1). Then $K \preceq L$ because we chose L to be maximal among all literals in C such that $L\theta = L_0$.

For *strict* maximality, we simply observe that if L occurs more than once in C, it also occurs more than once in $C\theta$. □

Lemma 4.28 (Lifting of Eligibility)**.** *Let $N \subseteq \mathcal{C}_{\mathrm{H}}$. Let $C_{\mathrm{G}} \in \mathcal{G}(N)$, let $C_{\mathrm{H}} = \mathcal{G}_N^{-1}(C_{\mathrm{G}})$, and let θ be a grounding substitution such that $C_{\mathrm{G}} = C_{\mathrm{H}}\theta$.*

- *Let L_{G} be a literal in C_{G} that is (strictly) eligible w.r.t. $\mathcal{G}(hsel, N)$. Then there exists a literal L_{H} in C_{H} such that $L_{\mathrm{G}} = L_{\mathrm{H}}\theta$ and, given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in C_{H}, L_{H} is (strictly) eligible in C_{H} w.r.t. σ and hsel.*
- *Let $L_{\mathrm{G}}.s_{\mathrm{G}}.p_{\mathrm{G}}$ be a green position of C_{G} that is eligible w.r.t. $\mathcal{G}(hsel, N)$. Then there exists a green position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ of C_{H} such that*
  - *$L_{\mathrm{G}} = L_{\mathrm{H}}\theta$;*
  - *$s_{\mathrm{G}} = s_{\mathrm{H}}\theta$;*
  - *$*$ $p_{\mathrm{G}} = p_{\mathrm{H}}$, or*
    - *$p_{\mathrm{G}} = p_{\mathrm{H}}.q$ for some nonempty q, the subterm u_{H} at position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ of C_{H} is variable-headed, and $u_{\mathrm{H}}\theta$ is nonfunctional; and*
  - *given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in C_{H}, $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ is eligible in C_{H} w.r.t. σ and hsel.*

Proof. Let L_{G} be a literal in C_{G} that is (strictly) eligible w.r.t. $\mathcal{G}(hsel, N)$. By the definition of eligibility (Definition 2.15), there are two ways to be (strictly) eligible:

- L_{G} is selected by $\mathcal{G}(hsel, N)$. By Definition 4.25, there exists a literal L_{H} selected by *hsel* such that $L_{\mathrm{G}} = L_{\mathrm{H}}\theta$. By Definition 2.15, L_{H} is (strictly) eligible in C_{H} w.r.t. σ because it is selected.
- There are no selected literals in C_{G} and L_{G} is (strictly) maximal in C_{G}. By Definition 4.25, there are no selected literals in C_{H}. Since $C_{\mathrm{G}} = C_{\mathrm{H}}\theta = C_{\mathrm{H}}\sigma\zeta$, by Lemma 4.27, there exists a literal $L_{\mathrm{H}} \in C_{\mathrm{H}}$ such that $L_{\mathrm{H}}\sigma$ is (strictly) maximal in $C_{\mathrm{H}}\sigma$. By Definition 2.15, L_{H} is (strictly) eligible in $C_{\mathrm{H}}[\![S]\!]$ w.r.t. σ.

For the second part of the lemma, let $L_{\mathrm{G}}.s_{\mathrm{G}}.p_{\mathrm{G}}$ be a green position of C_{G} that is eligible w.r.t. $\mathcal{G}(hsel, N)$. By Definition 2.15, the literal L_{G} is of the form $s_{\mathrm{G}} \mathrel{\dot{\approx}} t_{\mathrm{G}}$ with $s_{\mathrm{G}} \succ t_{\mathrm{G}}$ and L_{G} is either negative and eligible or positive and strictly eligible. By the first part of this lemma, there exists a literal L_{H} in C_{H} that is either negative and eligible or positive and strictly eligible in C_{H} w.r.t. σ and *hsel* such that $L_{\mathrm{G}} = L_{\mathrm{H}}\theta$. Then L_{H} must be of the form $s_{\mathrm{H}} \mathrel{\dot{\approx}} t_{\mathrm{H}}$ with $s_{\mathrm{G}} = s_{\mathrm{H}}\theta$ and $t_{\mathrm{G}} = t_{\mathrm{H}}\theta$. Since $s_{\mathrm{G}} \succ t_{\mathrm{G}}$, we have $s_{\mathrm{H}} \not\preceq t_{\mathrm{H}}$. By Definition 2.15, every green position in $L_{\mathrm{H}}.s_{\mathrm{H}}$ is eligible in C_{H} w.r.t. σ and *hsel*.

It remains to show that there exists a green position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ in C_{H} such that either $p_{\mathrm{G}} = p_{\mathrm{H}}$ or $p_{\mathrm{G}} = p_{\mathrm{H}}.q$ for some nonempty q, the subterm u_{H} at position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ of C_{H} is variable-headed, and $u_{\mathrm{H}}\theta$ is nonfunctional.

Since p_{G} is a green position of $s_{\mathrm{G}} = s_{\mathrm{H}}\theta$, position p_{G} must either be a green position of s_{H} or be below a variable-headed term in s_{H}. In the first case, we set $p_{\mathrm{H}} = p_{\mathrm{G}}$. In the second case, let u_{H} be that variable-headed term and let p_{H} be its position. Then $p_{\mathrm{H}}.q = p_{\mathrm{G}}$ for some nonempty q. Moreover, since p_{G} is a green position of s_{G}, the subterm of s_{G} at position p_{H}, which is $u_{\mathrm{H}}\theta$, cannot be functional. □

Lemma 4.29 (Lifting Lemma)**.** *Let $N \subseteq \mathcal{C}_{\mathrm{H}}$ be saturated up to redundancy w.r.t. $HInf^{\succ,hsel}$. Then $\mathcal{G}(N)$ is saturated up to redundancy w.r.t. $GInf^{\succ,\mathcal{G}(hsel,N)}$.*

Proof. Let ι_{G} be a *GInf* inference from $\mathcal{G}(N)$. We must show that $\iota_{\mathrm{G}} \in \mathit{GRed}_{\mathrm{I}}(\mathcal{G}(N))$. By Definitions 4.22, 4.15, 4.11, and Lemma 4.5, it suffices to show that

$$\begin{aligned} \mathcal{F}(\mathcal{G}(N)) &\models_{\mathrm{o}\lambda} \mathcal{F}(\mathit{concl}(\iota_{\mathrm{G}})) && \text{if } \iota_{\mathrm{G}} \text{ is a DIFF inference, or} \\ \{E \in \mathcal{F}(\mathcal{G}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(\mathit{mprem}(\iota_{\mathrm{G}}))\} &\models_{\mathrm{o}\lambda} \mathcal{F}(\mathit{concl}(\iota_{\mathrm{G}})) && \text{if } \iota_{\mathrm{G}} \text{ is some other inference.} \end{aligned} \tag{$*$}$$

One strategy that we will apply below is to construct an inference ι_{H} such that the prefixes of ι_{H} and ι_{G} match up to the prefixes G and FLUID and to construct substitutions $\theta_1, \ldots, \theta_{m+1}$ such that applying $\theta_1, \ldots, \theta_m$ to $\mathit{prems}(\iota_{\mathrm{H}})$ yields $\mathit{prems}(\iota_{\mathrm{G}})$ and applying θ_{m+1} to $\mathit{concl}(\iota_{\mathrm{H}})$ yields $\mathit{concl}(\iota_{\mathrm{G}})$. $(**)$

By Lemmas 4.21 and 4.14, $\mathcal{F}(\mathcal{J}(\iota_{\mathrm{G}}))$ is a valid $\mathit{FInf}^{\succ_{\mathcal{JF}}, \mathcal{F}(\mathcal{J}(\mathcal{G}(\mathit{hsel}, N)))}$ inference, By Lemma 4.5, $\succ_{\mathcal{JF}} = \succ_{\mathcal{F}}$ and $\mathcal{F}(\mathcal{J}(\mathcal{G}(\mathit{hsel}, N))) = \mathcal{F}(\mathcal{G}(\mathit{hsel}, N))$. Moreover, Lemma 4.5 tells us that $\mathcal{F}(\mathcal{J}(\iota_{\mathrm{G}}))$ can also be obtained by applying $\mathcal{F}$ directly to premisses and conclusion of ι_{G}. Therefore, ι_{H} is rooted in *FInf* for $(\theta_1, \ldots, \theta_{m+1})$ (Definition 2.24). By saturation of N up to redundancy w.r.t. *HInf*, ι_{H} is redundant and thus $(*)$ by Definition 2.25.

GSUP: Assume that ι_{G} is a GSUP inference

$$\frac{\overbrace{D'_{\mathrm{G}} \vee t_{\mathrm{G}} \approx t'_{\mathrm{G}}}^{D_{\mathrm{G}}} \quad C_{\mathrm{G}}\langle t_{\mathrm{G}} \rangle}{D'_{\mathrm{G}} \vee C_{\mathrm{G}}\langle t'_{\mathrm{G}} \rangle}\,\mathrm{IGSUP}$$

Let $C_{\mathrm{H}} = \mathcal{G}_N^{-1}(C_{\mathrm{G}})$, and let θ be a grounding substitution such that $C_{\mathrm{G}} = C_{\mathrm{H}}\theta$. By condition 4 of GSUP, the position $L_{\mathrm{G}}.s_{\mathrm{G}}.p_{\mathrm{G}}$ of t_{G} is eligible in C_{G}. By Lemma 4.28, there exists a green position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ of C_{H} such that

- $L_{\mathrm{G}} = L_{\mathrm{H}}\theta$;
- $s_{\mathrm{G}} = s_{\mathrm{H}}\theta$;
- given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in C_{H}, $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ is eligible in C_{H} w.r.t. σ and *hsel* (condition 5 of SUP or FLUIDSUP);

and one of the following cases applies:

CASE 1: $p_{\mathrm{G}} = p_{\mathrm{H}}$.

CASE 1A: The subterm at position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ of C_{H} is a variable x and no occurrence of x is inside of a parameter in C_{H}.

Since free De Bruijn indices cannot occur in parameters, x does not occur in parameters in $C_{\mathrm{H}}\theta[x \mapsto x]$ either. By condition 1 of GSUP, t_{G} is nonfunctional and so x is nonfunctional. Thus, all occurrences of x in $C_{\mathrm{H}}\theta[x \mapsto x]$ are in green positions. Thus, $C_{\mathrm{H}}\theta[x \mapsto t'_{\mathrm{G}}]$ results from $C_{\mathrm{H}}\theta = C_{\mathrm{G}}$ by replacing t_{G} with t'_{G} at green positions. Thus, by the $\mathcal{T}_{\mathrm{G}}$-analogue of Lemma 4.3, $\mathcal{F}(C_{\mathrm{G}})$ results from $\mathcal{F}(C_{\mathrm{H}}\theta[x \mapsto t'_{\mathrm{G}}])$ by replacing $\mathcal{F}(t_{\mathrm{G}})$ with $\mathcal{F}(t'_{\mathrm{G}})$. Therefore, $\{\mathcal{F}(t_{\mathrm{G}}) \approx \mathcal{F}(t'_{\mathrm{G}}), \mathcal{F}(C_{\mathrm{H}}\theta[x \mapsto t'_{\mathrm{G}}])\} \models_{\mathrm{o}\lambda} \mathcal{F}(C_{\mathrm{G}})$ and thus

$$\{\mathcal{F}(D_{\mathrm{G}}), \mathcal{F}(C_{\mathrm{H}}\theta[x \mapsto t'_{\mathrm{G}}])\} \models_{\mathrm{o}\lambda} \mathcal{F}(D'_{\mathrm{G}} \vee C_{\mathrm{G}}\langle t'_{\mathrm{G}} \rangle)$$

By condition 3 of GSUP, $D_{\mathrm{G}} \prec \mathit{mprem}(\iota_{\mathrm{G}})$, and thus $\mathcal{F}(D_{\mathrm{G}}) \prec_{\mathcal{F}} \mathcal{F}(\mathit{mprem}(\iota_{\mathrm{G}}))$. By Condition 2 of GSUP, $t_{\mathrm{G}} \succ t'_{\mathrm{G}}$, and thus by $(\mathrm{O2})_{\mathrm{F}}$, $\mathcal{F}(\mathit{mprem}(\iota_{\mathrm{G}})) = \mathcal{F}(C_{\mathrm{H}}\theta[x \mapsto t_{\mathrm{G}}]) \succ_{\mathcal{F}} \mathcal{F}(C_{\mathrm{H}}\theta[x \mapsto t'_{\mathrm{G}}])$. Therefore,

$$\{E \in \mathcal{F}(\mathcal{G}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(\mathit{mprem}(\iota_{\mathrm{G}}))\} \models_{\mathrm{o}\lambda} \mathcal{F}(\mathit{concl}(\iota_{\mathrm{G}}))$$

and thus $(*)$.

Case 1b: The subterm u_H at position $L_\mathrm{H}.s_\mathrm{H}.p_\mathrm{H}$ of C_H is not a variable or there exists an occurrence of that variable inside of a parameter in C_H.

Then we construct a Sup inference ι_H from $D_\mathrm{H} = \mathcal{G}_N^{-1}(D_\mathrm{G})$, and $C_\mathrm{H} = \mathcal{G}_N^{-1}(C_\mathrm{G})$. Let ρ be the grounding substitution such that $D_\mathrm{G} = D_\mathrm{H}\rho$ and let θ be the grounding substitution such that $C_\mathrm{G} = C_\mathrm{H}\theta$.

The assumption of this cases matches exactly condition 2 of Sup.

Condition 5 of GSup states that $t_\mathrm{G} \approx t'_\mathrm{G}$ is strictly eligible in D_G. Let $D_\mathrm{H} = D'_\mathrm{H} \vee t_\mathrm{H} \approx t'_\mathrm{H}$, where $t_\mathrm{H} \approx t'_\mathrm{H}$ is the literal that Lemma 4.28 guarantees to be strictly eligible w.r.t. any suitable σ (condition 6 of Sup), with $t_\mathrm{H}\rho = t_\mathrm{G}$ and $t'_\mathrm{H}\rho = t'_\mathrm{G}$. Condition 6 of GSup states that if t'_G is Boolean, then $t'_\mathrm{G} = \mathsf{T}$. Thus, there are no selected literals in D_G. By Definition 4.25, it follows that there are no selected literals in D_H (condition 7 of Sup).

Since $L_\mathrm{G} = L_\mathrm{H}\theta$, $s_\mathrm{G} = s_\mathrm{H}\theta$, and $p_\mathrm{G} = p_\mathrm{H}$, we have $u_\mathrm{H}\theta = t_\mathrm{G} = t_\mathrm{H}\rho$. Since the variables of D_H and C_H are disjoint, there exists a subsitution $\theta \cup \rho$ that matches θ on all variables of C_H and ρ on all variables of D_H. This substitution is a unifier of $t_\mathrm{H} \equiv u_\mathrm{H}$. By Definition 2.11, there exists a unifier $\sigma \in \mathrm{CSU}(t_\mathrm{H} \equiv u_\mathrm{H})$ and a substitution ζ such that $x\sigma\zeta = x(\theta \cup \rho)$ for all variables x in C_H or D_H (condition 1 of Sup).

Since t_G is nonfunctional by condition 1 of GSup, and since $u_\mathrm{H}\sigma\zeta = u_\mathrm{H}\theta = t_\mathrm{G}$, $u_\mathrm{H}\sigma$ is nonfunctional (condition 3 of Sup).

By condition 2 of GSup, $t_\mathrm{H}\sigma\zeta = t_\mathrm{G} \succ t'_\mathrm{G} = t'_\mathrm{H}\sigma\zeta$, and thus by Lemma 4.26, $t_\mathrm{H}\sigma \not\trianglerighteq t'_\mathrm{H}\sigma$ (condition 4 of Sup).

Moreover, $\mathit{concl}(\iota_\mathrm{H})\zeta = \mathit{concl}(\iota_\mathrm{G})$. Thus, we can apply $(**)$.

Case 2: $p_\mathrm{G} = p_\mathrm{H}.q$ for some nonempty q, the subterm u_H at position $L_\mathrm{H}.s_\mathrm{H}.p_\mathrm{H}$ of C_H is variable-headed, and $u_\mathrm{H}\theta$ is nonfunctional.

Case 2a: The subterm at position $L_\mathrm{H}.s_\mathrm{H}.p_\mathrm{H}$ of C_H is a variable and no occurrence of that variable is inside of a parameter in C_H.

Then we can proceed as in Case 1a.

Case 2b: The subterm u_H at position $L_\mathrm{H}.s_\mathrm{H}.p_\mathrm{H}$ of C_H is not a variable or there exists an occurrence of that variable inside of a parameter in C_H.

Then we construct a FluidSup inference ι_H from $D_\mathrm{H} = \mathcal{G}_N^{-1}(D_\mathrm{G})$, and $C_\mathrm{H} = \mathcal{G}_N^{-1}(C_\mathrm{G})$. Let ρ be the grounding substitution such that $D_\mathrm{G} = D_\mathrm{H}\rho$ and let θ be the grounding substitution such that $C_\mathrm{G} = C_\mathrm{H}\theta$.

The assumptions of this case imply condition 2 of FluidSup. We define D'_H, t_H, and t'_H in the same way as in Case 1b. This ensures condition 6 and condition 7 of FluidSup.

Let z be a fresh variable (condition 8 of FluidSup). Let $v = \lambda\,(u_\mathrm{H}\theta)\langle n\rangle_q$, where n is the appropriate De Bruijn index to refer to the initial λ. We define θ' by $z\theta' = v$, $x\theta' = x\rho$ for all variables x in D_H and $x\theta' = x\theta$ for all other variables x. Then, $(z\,t_\mathrm{H})\theta' = v\,(t_\mathrm{H}\rho) = v\,t_\mathrm{G} = (u_\mathrm{H}\theta)\langle t_\mathrm{G}\rangle_q = u_\mathrm{H}\theta = u_\mathrm{H}\theta'$. So θ' is a unifier of $z\,t_\mathrm{H}$ and u_H. Thus, by definition of CSU (Definition 2.11), there exists a unifier $\sigma \in \mathrm{CSU}(z\,t_\mathrm{H} \equiv u_\mathrm{H})$ and a substitution ζ such that $x\sigma\zeta = x\theta'_\mathrm{H}$ for all relevant variables x (condition 1 of FluidSup).

The assumption of Case 2 tells us that $u_\mathrm{H}\theta = u_\mathrm{H}\sigma\zeta$ is nonfunctional. It follows that $u_\mathrm{H}\sigma$ is nonfunctional (condition 3 of FluidSup).

By condition 2 of GSup, $t_\mathrm{H}\sigma\zeta = t_\mathrm{G} \succ t'_\mathrm{G} = t'_\mathrm{H}\sigma\zeta$, and thus by Lemma 4.26, $t_\mathrm{H}\sigma \not\trianglerighteq t'_\mathrm{H}\sigma$ (condition 4 of FluidSup).

By condition 2 of GSup, $t_\mathrm{H}\rho = t_\mathrm{G} \neq t'_\mathrm{G} = t'_\mathrm{H}\rho$. Thus, $(z t_\mathrm{H})\sigma\zeta = v(t_\mathrm{H}\rho) = u_\mathrm{H}\theta\langle t_\mathrm{H}\rho\rangle_q \neq u_\mathrm{H}\theta\langle t'_\mathrm{H}\rho\rangle_q = v\,(t'_\mathrm{H}\rho) = (z\,t'_\mathrm{H})\sigma\zeta$. So, $(z\,t'_\mathrm{H})\sigma \neq (z\,t_\mathrm{H})\sigma$ (condition 9 of FluidSup).

Since $z\sigma\zeta = v$ and $v \neq \lambda\, 0$ because q is nonempty, we have $z\sigma \neq \lambda\, 0$ (condition 10 of FluidSup).

Moreover, $\mathit{concl}(\iota_{\mathrm{H}})\zeta = (D'_{\mathrm{H}} \vee C_{\mathrm{H}}\langle z\ t'_{\mathrm{H}}\rangle_{p_{\mathrm{H}}})\sigma\zeta = D'_{\mathrm{G}} \vee C_{\mathrm{G}}\langle (u_{\mathrm{H}}\theta)\langle t'_{\mathrm{G}}\rangle_q\rangle_{p_{\mathrm{H}}} = D'_{\mathrm{G}} \vee C_{\mathrm{G}}\langle t'_{\mathrm{G}}\rangle_{p_{\mathrm{G}}} = \mathit{concl}(\iota_{\mathrm{G}})$. Thus, we can apply $(**)$.

GEqRes: Assume that ι_{G} is a GEqRes inference

$$\frac{\overbrace{C'_{\mathrm{G}} \vee u_{\mathrm{G}} \not\approx u_{\mathrm{G}}}^{C_{\mathrm{G}}}}{C'_{\mathrm{G}}}\,\mathrm{GEqRes}$$

Let $C_{\mathrm{H}} = \mathcal{G}_N^{-1}(C_{\mathrm{G}})$, and let θ be a grounding substitution such that $C_{\mathrm{G}} = C_{\mathrm{H}}\theta$. Condition 1 of GEqRes states that $u_{\mathrm{G}} \not\approx u_{\mathrm{G}}$ is strictly eligible in C_{G}. Let $C_{\mathrm{H}} = C'_{\mathrm{H}} \vee u_{\mathrm{H}} \not\approx u'_{\mathrm{H}}$, where $u_{\mathrm{H}} \not\approx u'_{\mathrm{H}}$ is the literal that Lemma 4.28 guarantees to be strictly eligible w.r.t. any suitable σ (condition 2 of GEqRes), with $u_{\mathrm{H}}\theta = u_{\mathrm{G}}$ and $u'_{\mathrm{H}}\theta = u_{\mathrm{G}}$. Then θ is a unifier of u_{H} and u'_{H}, and thus there exists a unifier $\sigma \in \mathrm{CSU}(u_{\mathrm{H}} \equiv u'_{\mathrm{H}})$ and a substitution ζ such that $x\sigma\zeta = x\theta$ for all variables x in C_{H} (condition 1 of GEqRes).

Moreover, $\mathit{concl}(\iota_{\mathrm{H}})\zeta = \mathit{concl}(\iota_{\mathrm{G}})$. Thus, we can apply $(**)$.

GEqFact: Assume that ι_{G} is a GEqFact inference

$$\frac{\overbrace{C'_{\mathrm{G}} \vee u_{\mathrm{G}} \approx v'_{\mathrm{G}} \vee u_{\mathrm{G}} \approx v_{\mathrm{G}}}^{C_{\mathrm{G}}}}{C'_{\mathrm{G}} \vee v_{\mathrm{G}} \not\approx v'_{\mathrm{G}} \vee u_{\mathrm{G}} \approx v_{\mathrm{G}}}\,\mathrm{GEqFact}$$

Let $C_{\mathrm{H}} = \mathcal{G}_N^{-1}(C_{\mathrm{G}})$, and let θ be a grounding substitution such that $C_{\mathrm{G}} = C_{\mathrm{H}}\theta$. Condition 1 of GEqFact states that $u_{\mathrm{G}} \approx v_{\mathrm{G}}$ is maximal in C_{G}. Let $u_{\mathrm{H}} \approx v_{\mathrm{H}}$ be the literal in C_{H} that Lemma 4.27 guarantees to be strictly maximal w.r.t. any suitable σ (condition 2 of EqFact), with $u_{\mathrm{H}}\theta = u_{\mathrm{G}}$ and $v_{\mathrm{H}}\theta = v_{\mathrm{G}}$. Choose C'_{H}, u'_{H}, and v'_{H} such that $C_{\mathrm{H}} = C'_{\mathrm{H}} \vee u'_{\mathrm{H}} \approx v'_{\mathrm{H}} \vee u_{\mathrm{H}} \approx v_{\mathrm{H}}$, $C'_{\mathrm{H}}\theta = C'_{\mathrm{G}}$, $u'_{\mathrm{H}}\theta = u_{\mathrm{G}}$, and $v'_{\mathrm{H}}\theta = v'_{\mathrm{G}}$.

Then θ is a unifier of u_{H} and u'_{H}, and thus there exists a unifier $\sigma \in \mathrm{CSU}(u_{\mathrm{H}} \equiv u'_{\mathrm{H}})$ and a substitution ζ such that $x\sigma\zeta = x\theta$ for all variables x in C_{H} (condition 1 of EqFact).

By condition 2 of GEqFact, there are no selected literals in C_{G} and thus there are no selected literals in C_{H} (condition 3 of EqFact).

By condition 3 of GEqFact, $u_{\mathrm{H}}\sigma\zeta = u_{\mathrm{G}} \succ v_{\mathrm{G}} = v_{\mathrm{H}}\sigma\zeta$. By Lemma 4.26, $u_{\mathrm{H}}\sigma \not\preceq v_{\mathrm{H}}\sigma$ (condition 4 of EqFact).

Moreover, $\mathit{concl}(\iota_{\mathrm{H}})\zeta = \mathit{concl}(\iota_{\mathrm{G}})$. Thus, we can apply $(**)$.

GClausify: Assume ι_{G} is a GClausify inference

$$\frac{\overbrace{C'_{\mathrm{G}} \vee s_{\mathrm{G}} \approx t_{\mathrm{G}}}^{C_{\mathrm{G}}}}{C'_{\mathrm{G}} \vee D_{\mathrm{G}}}\,\mathrm{GClausify}$$

with τ_{G} being the type and u_{G} and v_{G} being the terms used for condition 2. Let $C_{\mathrm{H}} = \mathcal{G}_N^{-1}(C_{\mathrm{G}})$, and let θ be a grounding substitution such that $C_{\mathrm{G}} = C_{\mathrm{H}}\theta$.

Condition 1 of GClausify is that $s_{\mathrm{G}} \approx t_{\mathrm{G}}$ is strictly eligible in C_{G}. Let $C_{\mathrm{H}} = C'_{\mathrm{H}} \vee s_{\mathrm{H}} \approx t_{\mathrm{H}}$, where $s_{\mathrm{H}} \approx t_{\mathrm{H}}$ is the literal that Lemma 4.28 guarantees to be strictly eligible w.r.t. any suitable σ (condition 2 of Clausify), with $s_{\mathrm{H}}\theta = s_{\mathrm{G}}$ and $t_{\mathrm{H}}\theta = t_{\mathrm{G}}$.

We distinguish two cases:

Case 1: s_{H} is a variable and no occurrence of that variable is inside of a parameter in C_{H}.

Then we define substitutions θ_{T} and $\theta_{\bot}$ that coincide with θ, except for mapping s_{H} to T and $\bot$, respectively. Since the semantics of $\models_{\mathrm{o}\lambda}$ interpret Booleans, we have $\{\mathcal{F}(C_{\mathrm{H}}\theta_{\mathsf{T}}), \mathcal{F}(C_{\mathrm{H}}\theta_{\bot})\} \models_{\mathrm{o}\lambda} \mathcal{F}(C_{\mathrm{H}}\theta)$ because s_{H} does not appear in parameters in C_{H}. By $(\mathrm{O4})_{\mathrm{F}}$ and $(\mathrm{O2})_{\mathrm{F}}$, $\mathcal{F}(C_{\mathrm{H}}\theta_{\mathsf{T}}) \prec \mathcal{F}(C_{\mathrm{H}}\theta_{\bot}) \prec \mathcal{F}(C_{\mathrm{H}}\theta)$. Therefore, we can apply $(*)$.

Case 2: s_{H} is not a variable or there exists an occurrence of that variable inside of a parameter in C_{H} (condition 3 of Clausify). Then we construct a corresponding Clausify inference ι_{H}.

Comparing the listed triples in GClausify and Clausify, we see that there must be a triple $(s'_{\mathrm{H}}, t'_{\mathrm{H}}, D_{\mathrm{H}})$ listed for Clausify such that $(s'_{\mathrm{H}}\rho, t'_{\mathrm{H}}\rho, D_{\mathrm{H}}\rho) = (s_{\mathrm{G}}, t_{\mathrm{G}}, D_{\mathrm{G}})$ with $\rho = \{\alpha \mapsto \tau_{\mathrm{G}}, x \mapsto u_{\mathrm{G}}, y \mapsto v_{\mathrm{G}}\}$ is the triple used for ι_{G} (condition 4 of Clausify).

Moreover, we observe that $s_{\mathrm{H}}\theta = s_{\mathrm{G}} = s'_{\mathrm{H}}\rho$ and $t_{\mathrm{H}}\theta = t_{\mathrm{G}} = t'_{\mathrm{H}}\rho$. Thus the substitution θ' mapping all variables x in s'_{H} and t'_{H} to $x\rho$ and all other variables x to $x\theta$ is a unifier of $s_{\mathrm{H}} \equiv s'_{\mathrm{H}}$ and $t_{\mathrm{H}} \equiv t'_{\mathrm{H}}$. So there exists a unifier $\sigma \in \mathrm{CSU}(s_{\mathrm{H}} \equiv s'_{\mathrm{H}}, t_{\mathrm{H}} \equiv t'_{\mathrm{H}})$ (condition 1 of Clausify) and a substitution ζ such that $x\sigma\zeta = x\theta'$ for all variables x in C_{H}.

Moreover, it follows that $\mathit{concl}(\iota_{\mathrm{H}})\zeta = \mathit{concl}(\iota_{\mathrm{G}})$. Thus, we can apply $(**)$.

GBoolHoist: Analogous to GSup, using the substitutions θ_{T} and $\theta_{\bot}$ as in Case 1 of GClausify.

GLoobHoist: Analogous to GBoolHoist.

GFalseElim: Analogous to GEqRes.

GArgCong: Analogous to GEqRes.

GExt: Assume that ι_{G} is a GExt inference

$$\frac{C_{\mathrm{G}}\langle u_{\mathrm{G}}\rangle}{C_{\mathrm{G}}\langle w_{\mathrm{G}}\rangle \vee u_{\mathrm{G}}\ \mathsf{diff}\langle\tau_{\mathrm{G}}, v_{\mathrm{G}}\rangle(u_{\mathrm{G}}, w_{\mathrm{G}}) \not\approx w_{\mathrm{G}}\ \mathsf{diff}\langle\tau_{\mathrm{G}}, v_{\mathrm{G}}\rangle(u_{\mathrm{G}}, w_{\mathrm{G}})}\ \mathrm{IGExt}$$

Let $C_{\mathrm{H}} = \mathcal{G}_N^{-1}(C_{\mathrm{G}})$, and let θ be a grounding substitution such that $C_{\mathrm{G}} = C_{\mathrm{H}}\theta$. By condition 1 of GExt, the position $L_{\mathrm{G}}.s_{\mathrm{G}}.p_{\mathrm{G}}$ of u_{G} is eligible in C_{G}. By Lemma 4.28, there exists a green position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ of C_{H} such that

- $L_{\mathrm{G}} = L_{\mathrm{H}}\theta$;
- $s_{\mathrm{G}} = s_{\mathrm{H}}\theta$;
- given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in C_{H}, $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ is eligible in C_{H} w.r.t. σ and *hsel* (condition 3 of Ext/condition 7 of FluidExt);

and one of the following cases applies:

Case 1: $p_{\mathrm{G}} = p_{\mathrm{H}}$. Then we construct an Ext inference ι_{H} from C_{H}.

Let u_{H} be the subterm of C_{H} at position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$. Since $p_{\mathrm{G}} = p_{\mathrm{H}}$, we have $u_{\mathrm{H}}\theta = u_{\mathrm{G}}$. By condition 2 of GExt, u_{G} is functional and thus there exists a most general type substitution σ such that $u_{\mathrm{H}}\sigma$ is functional (condition 1 of Ext). By definition of 'most general', there exists a substitution ζ such that $x\sigma\zeta = x\theta$ for all variables x in C_{H}.

Let y be a fresh variable of the same type as $u_{\mathrm{H}}\sigma$ (condition 2 of Ext). Let $\zeta' = \zeta[y \mapsto w_{\mathrm{G}}]$. Then, $\mathit{concl}(\iota_{\mathrm{H}})\zeta' = \mathit{concl}(\iota_{\mathrm{G}})$. Thus, we can apply $(**)$.

Case 2: $p_{\mathrm{G}} = p_{\mathrm{H}}.q$ for some nonempty q, the subterm u_{H} at position $L_{\mathrm{H}}.s_{\mathrm{H}}.p_{\mathrm{H}}$ of C_{H} is variable-headed, and $u_{\mathrm{H}}\theta$ is nonfunctional.

Then we construct a FluidSup inference ι_{H} from $C_{\mathrm{H}} = \mathcal{G}_N^{-1}(C_{\mathrm{G}})$.

The assumption of this case implies condition 1 of FluidExt.

Let α and β be fresh type variables. Let x and y be fresh variables of type $\alpha \to \beta$, and let z be a fresh variable of function type from $\alpha \to \beta$ to the type of u_{H} (condition 3 of FluidExt).

Let $v = \lambda\,(u_{\mathrm{H}}\theta)\langle n\rangle_q$, where n is the appropriate De Bruijn index to refer to the initial λ. We define a substitutionθ' that coincides with θ on all variables except $\alpha\theta' = \tau_{\mathrm{G}}$, $\beta\theta' = \upsilon_{\mathrm{G}}$, $z\theta' = v$, $x\theta' = u_{\mathrm{G}}$, and $y\theta' = w_{\mathrm{G}}$. Then, $(z\,x)\theta' = v\,u_{\mathrm{G}} = (u_{\mathrm{H}}\theta)\langle u_{\mathrm{G}}\rangle_q = u_{\mathrm{H}}\theta = u_{\mathrm{H}}\theta'$. So θ' is a unifier of $z\,x$ and u_{H}. Thus, by definition of CSU (Definition 2.11), there exists a unifier $\sigma \in \mathrm{CSU}(z\,x \equiv u_{\mathrm{H}})$ and a substitution ζ such that $x\sigma\zeta = x\theta'$ for all relevant variables x (condition 4 of FluidExt).

The assumption of Case 2 tells us that $u_{\mathrm{H}}\theta = u_{\mathrm{H}}\sigma\zeta$ is nonfunctional. It follows that $u_{\mathrm{H}}\sigma$ is nonfunctional (condition 2 of FluidExt).

By condition 4 of GExt, $u_{\mathrm{G}} \neq w_{\mathrm{G}}$. Thus, $(z\,x)\sigma\zeta = v\,u_{\mathrm{G}} = (u_{\mathrm{H}}\theta)\langle u_{\mathrm{G}}\rangle_q \neq (u_{\mathrm{H}}\theta)\langle w_{\mathrm{G}}\rangle_q = v\,w_{\mathrm{G}} = (z\,y)\sigma\zeta$. So, $(z\,x)\sigma \neq (z\,y)\sigma$ (condition 5 of FluidExt).

Since $z\sigma\zeta = v$ and $v \neq \lambda\,0$ because q is nonempty, we have $z\sigma \neq \lambda\,0$ (condition 6 of FluidExt).

Moreover,

$$\begin{aligned}
\mathit{concl}(\iota_{\mathrm{H}})\zeta &= (C_{\mathrm{H}}\langle z\,y\rangle \vee x\,\mathsf{diff}\langle\alpha,\beta\rangle(x,y) \not\approx y\,\mathsf{diff}\langle\alpha,\beta\rangle(x,y))\sigma\zeta \\
&= C_{\mathrm{G}}\langle(u_{\mathrm{H}}\theta)\langle w_{\mathrm{G}}\rangle_q\rangle_{p_{\mathrm{H}}} \vee u_{\mathrm{G}}\,\mathsf{diff}\langle\tau_{\mathrm{G}},\upsilon_{\mathrm{G}}\rangle(u_{\mathrm{G}},w_{\mathrm{G}}) \not\approx w_{\mathrm{G}}\,\mathsf{diff}\langle\tau_{\mathrm{G}},\upsilon_{\mathrm{G}}\rangle(u_{\mathrm{G}},w_{\mathrm{G}}) \\
&= C_{\mathrm{G}}\langle w_{\mathrm{G}}\rangle_{p_{\mathrm{G}}} \vee u_{\mathrm{G}}\,\mathsf{diff}\langle\tau_{\mathrm{G}},\upsilon_{\mathrm{G}}\rangle(u_{\mathrm{G}},w_{\mathrm{G}}) \not\approx w_{\mathrm{G}}\,\mathsf{diff}\langle\tau_{\mathrm{G}},\upsilon_{\mathrm{G}}\rangle(u_{\mathrm{G}},w_{\mathrm{G}}) \\
&= \mathit{concl}(\iota_{\mathrm{G}})
\end{aligned}$$

Thus, we can apply (∗∗).

GDiff: Assume that ι_{G} is a GDiff inference

$$\frac{}{u\,\mathsf{diff}\langle\tau,\upsilon\rangle(u,w) \not\approx u\,\mathsf{diff}\langle\tau,\upsilon\rangle(u,w) \vee u\,s \approx w\,s}\,\textsc{GDiff}$$

Then we use the following Diff inference ι_{H}:

$$\frac{}{y\,(\mathsf{diff}\langle\alpha,\beta\rangle(y,z)) \not\approx z\,(\mathsf{diff}\langle\alpha,\beta\rangle(y,z)) \vee y\,x \approx z\,x}\,\textsc{Diff}$$

Clearly, $\mathit{concl}(\iota_{\mathrm{H}})\theta = \mathit{concl}(\iota_{\mathrm{G}})$ for $\theta = \{\alpha \mapsto \tau, \beta \mapsto \upsilon, y \mapsto u, z \mapsto w, x \mapsto s\}$. Thus, we can apply (∗∗). □

4.4. Model Construction. In this subsection, we construct models of saturated clause sets, starting with a first-order model and lifting it through the levels. Using the results of Section 4.3, we prove a completeness property for each of the calculi that roughly states the following. For any saturated set N_∞ that does not contain the empty clause, there exists a model of N_∞.

Finally, in level H, we bring everything together by showing that the constructed model is also a model of N_0. It follows that the calculus *HInf* is refutationally complete.

4.4.1. *First-Order Levels.* In this subsubsection, let $\succ$ be an admissible term order for *FInf* (Definition 4.6), and let *fsel* be a selection function on $\mathcal{C}_{\mathrm{F}}$.

The completeness proof for *FInf* relies on constructing a first-order term rewrite system. For any first-order term rewrite system R, there exists a first-order interpretation, which we also denote R, such that $R \models_{\mathrm{fol}} s \approx t$ if and only if $s \leftrightarrow_R^* t$. Formally, this can be implemented by a first-order interpretation whose universe for each type τ consists of the R-equivalence classes of ground terms of type τ.

Definition 4.30 (R_N)**.** Let N be a set of ground first-order clauses with $\bot \notin N$. By well-founded induction, we define term rewrite systems R_e and Δ_e for all ground clauses and ground terms $e \in \mathcal{T}_{\mathrm{F}} \cup \mathcal{C}_{\mathrm{F}}$ and finally a term rewrite system R_N. As our well-founded order on $\mathcal{T}_{\mathrm{F}} \cup \mathcal{C}_{\mathrm{F}}$, we employ our term and clause order $\succ$. To compare terms with clauses, we define a term s to be larger than a clause C if and only if s is larger than every term in C. Formally, this can be defined using the clause order by Bachmair and Ganzinger [1, Sect. 2.4] and encoding a term s as the multiset $\{\{\{s\}\}\}$.

(Δ1) Logical Boolean rewrites: Given a term s, let $\Delta_s = \{s \to t\}$ if
- (s, t) is one of the following:

$(\neg\bot, \top)$	$(\top \wedge \top, \top)$	$(\top \vee \top, \top)$	$(\top \to \top, \top)$	$(u \approx^\tau u, \top)$
$(\neg\top, \bot)$	$(\top \wedge \bot, \bot)$	$(\top \vee \bot, \top)$	$(\top \to \bot, \bot)$	$(u \approx^\tau v, \bot)$ with $u \neq v$
	$(\bot \wedge \top, \bot)$	$(\bot \vee \top, \top)$	$(\bot \to \top, \top)$	$(u \not\approx^\tau u, \bot)$
	$(\bot \wedge \bot, \bot)$	$(\bot \vee \bot, \bot)$	$(\bot \to \bot, \top)$	$(u \not\approx^\tau v, \top)$ with $u \neq v$

- s is irreducible w.r.t. R_s.

(Δ2) Backstop Boolean rewrites: Given a clause C, let $\Delta_C = \{s \to \bot\}$ if
- $C = s \approx \bot$;
- $s \notin \{\bot, \top\}$;
- s is irreducible w.r.t. R_C.

(Δ3) Function rewrites: Given a clause C, let $\Delta_C = \{\mathcal{F}(u) \to \mathcal{F}(w)\}$ if
- $C = \mathcal{F}(u) \approx \mathcal{F}(w)$ for functional terms u and w;
- $\mathcal{F}(u) \succ \mathcal{F}(w)$
- $\mathcal{F}(u\,\mathsf{diff}_{s,t}^{\tau,v}) \leftrightarrow_{R_C}^* \mathcal{F}(v\,\mathsf{diff}_{s,t}^{\tau,v})$ for all s, t;
- $\mathcal{F}(u)$ is irreducible w.r.t. R_C.

(Δ4) Produced rewrites: Given a clause C, let $\Delta_C = \{s \to t\}$ if
- (CC1) $C = C' \vee s \approx t$ for some clause C' and terms s and t;
- (CC2) s is nonfunctional;
- (CC3) the root of s is not a logical symbol;
- (CC4) if t is Boolean, then $t = \top$
- (CC5) $s \succ t$;
- (CC6) $s \approx t$ is maximal in C;
- (CC7) there are no selected literals in C;
- (CC8) s is irreducible by R_C;
- (CC9) $R_C \not\models_{\mathrm{fol}} C$;
- (CC10) $R_C \cup \{s \to t\} \not\models_{\mathrm{fol}} C'$.

In this case, we say that C produces $s \to t$ and that C is *productive*.

(Δ5) For all other terms and clauses e, Let $\Delta_e = \emptyset$.

Let $R_e = \bigcup_{f \prec e} \Delta_f$. Let $R_N = \bigcup_{e \in \mathcal{T}_{\mathrm{F}} \cup \mathcal{C}_{\mathrm{F}}} \Delta_e$.

Lemma 4.31. *The rewrite systems R_C and R_N do not have critical pairs and are oriented by $\succ$.*

Proof. It is easy to check that all rules in R_C and R_N are oriented by $\succ$, using $(\mathrm{O4})_\mathrm{F}$.

To show the absence of critical pairs, suppose that there exists a critical pair $s \to t$ and $s' \to t'$ in R_N, originating from Δ_e and $\Delta_{e'}$ respectively, for some $e, e' \in \mathcal{T}_\mathrm{F} \cup \mathcal{C}_\mathrm{F}$. Without loss, we assume $e \succ e'$. Inspecting the rules of Definition 4.30, it follows that $s \succeq s'$. By the subterm property $(\mathrm{O3})_\mathrm{F}$, s cannot be a proper subterm of s'. So for the rules to be a critical pair, s' must be a subterm of s. But then s is not irreducible by $\Delta_{e'} \subseteq R_e$, contradicting the irreducibility conditions of Definition 4.30. □

Lemma 4.32. *The normal form of any ground Boolean term w.r.t. R_N is T or $\bot$.*

Proof. Inspecting the rules of Definition 4.30, in particular (CC3), we see that T and $\bot$ are irreducible w.r.t. R_N.

It remains to show that any ground Boolean term s reduces to T or $\bot$. We prove the claim by induction on s w.r.t. $\succ$. If $s = \mathsf{T}$ or $s = \bot$, we are done. Otherwise, consider the rule $(\Delta 2)$ for $C = s \approx \bot$. Either s is reducible by R_C or $(\Delta 2)$ triggers, making s reducible by Δ_C. In both cases, s is reducible by R_N. Let s' be the result of reducing s by R_N. By Lemma 4.31, $s \succ s'$. By the induction hypothesis, s' reduces to T or $\bot$. Therefore, s reduces to T or $\bot$. □

Lemma 4.33. *For all ground clauses C, if $R_C \models_\mathrm{fol} C$, then $R_N \models_\mathrm{fol} C$.*

Proof. We assume that $R_C \models_\mathrm{fol} C$. Then we have $R_C \models_\mathrm{fol} L$ for some literal L of C. It suffices to show that $R_N \models_\mathrm{fol} L$.

If $L = t \approx t'$ is a positive literal, then $t \leftrightarrow_{R_C}^* t'$. Since $R_C \subseteq R_N$, this implies $t \leftrightarrow_{R_N}^* t'$. Thus, $R_N \models_\mathrm{fol} L$.

If $L = t \not\approx t'$ is a negative literal, then $t \not\leftrightarrow_{R_C}^* t'$. By Lemma 4.31, this means that t and t' have different normal forms w.r.t. R_C. Without loss of generality, let $t \succ t'$. Let $s \mathrel{\dot\approx} s'$ be the maximal literal in C with $s \succeq s'$. We have $s \succ t$ if $s \mathrel{\dot\approx} s'$ is positive and $s \succeq t$ if $s \mathrel{\dot\approx} s'$ is negative. Hence, inspecting Definition 4.30, we see that the left-hand sides of rules in $\bigcup_{e \succeq C} \Delta_e$ are larger than t. Since only rules with a left-hand side smaller or equal to t can be involved in normalizing t and t' and $R_C \cup \bigcup_{e \succeq C} \Delta_e = R_N$, it follows that t and t' have different normal forms w.r.t. R_N. Therefore, $t \not\leftrightarrow_{R_N}^* t'$ and $R_N \models_\mathrm{fol} L$. □

Lemma 4.34. *If a clause $C = C' \vee s \approx t \in \mathcal{C}_\mathrm{F}$ produces $s \to t$, then $R_N \not\models_\mathrm{fol} C'$.*

Proof. By (CC5) and (CC6), all terms in C are smaller or equal to s. By (CC10), we have $R_C \cup \{s \to t\} \not\models C'$. The other rules $R_N \setminus (R_C \cup \{s \to t\})$ do not play any role in the truth of C because their left-hand sides are greater than s, as we can see by inspecting the rules of Definition 4.30, in particular the irreducibility conditions, and because R_N is confluent and terminating (Lemma 4.31). So, $R_C \cup \{s \to t\} \not\models_\mathrm{fol} C'$ implies $R_N \not\models_\mathrm{fol} C'$. □

Lemma 4.35. *Let u and w be higher-order ground terms of type $\tau \to \upsilon$. If $\mathcal{F}(u) \leftrightarrow_{R_N}^* \mathcal{F}(w)$, then $\mathcal{F}(u\,\mathsf{diff}_{s,t}^{\tau,\upsilon}) \leftrightarrow_{R_N}^* \mathcal{F}(w\,\mathsf{diff}_{s,t}^{\tau,\upsilon})$ for all s, t.*

Proof. By induction over each rewrite step in $\mathcal{F}(u) \leftrightarrow_{R_N}^* \mathcal{F}(w)$, it suffices to show the following claim: If $\mathcal{F}(u) \to_{R_N} \mathcal{F}(w)$, then $\mathcal{F}(u\,\mathsf{diff}_{s,t}^{\tau,\upsilon}) \leftrightarrow_{R_N}^* \mathcal{F}(w\,\mathsf{diff}_{s,t}^{\tau,\upsilon})$ for all s, t. Here, it is crucial that s and t are not necessarily equal to u and w.

By definition of $\mathcal{F}$, since u is functional, $\mathcal{F}(u) = \mathsf{fun}_u$. So $\mathcal{F}(u)$ has no proper subterms, and thus the rewrite step must happen at the root of $\mathcal{F}(u)$. Inspecting the definition of R_N,

we observe that the rewrite rule must originate from (Δ3). One of the conditions of (Δ3) then yields the claim. □

Lemma 4.36. *Let u and w be higher-order ground terms of type $\tau \to \upsilon$. If $\mathcal{F}(u\,\mathsf{diff}^{\tau,\upsilon}_{s,t}) \leftrightarrow^*_{R_N} \mathcal{F}(w\,\mathsf{diff}^{\tau,\upsilon}_{s,t})$ for all s, t, then $\mathcal{F}(u) \leftrightarrow^*_{R_N} \mathcal{F}(w)$.*

Proof. Let $\mathcal{F}(u') = \mathcal{F}(u)\downarrow_{R_N}$ and $\mathcal{F}(w') = \mathcal{F}(w)\downarrow_{R_N}$. By applying Lemma 4.35 to $\mathcal{F}(u) \leftrightarrow^*_{R_N} \mathcal{F}(u')$ and to $\mathcal{F}(w') \leftrightarrow^*_{R_N} \mathcal{F}(w)$, we have $\mathcal{F}(u'\,\mathsf{diff}^{\tau,\upsilon}_{s,t}) \leftrightarrow^*_{R_N} \mathcal{F}(w'\,\mathsf{diff}^{\tau,\upsilon}_{s,t})$ for all s, t.

We want to show that $\mathcal{F}(u) \leftrightarrow^*_{R_N} \mathcal{F}(w)$—i.e., that $\mathcal{F}(u') = \mathcal{F}(w')$. To derive a contradiction, we assume that $\mathcal{F}(u') \neq \mathcal{F}(w')$. Without loss of generality, we may assume that $\mathcal{F}(u') \succ \mathcal{F}(w')$. Then, using $(\mathrm{O5})_\mathrm{F}$, all conditions of (Δ3) are satisfied for the rule $\mathcal{F}(u') \to \mathcal{F}(w')$, contradicting the fact that $\mathcal{F}(u')$ is a normal form. □

Lemma 4.37. *R_N is a $\models_{\mathrm{o}\lambda}$-interpretation.*

Proof. We must prove all conditions listed in Section 2.6.

- By Lemma 4.32, the Boolean type has exactly two elements, namely the interpretations of T and $\bot$. The rule (Δ1) ensures that the symbols $\neg$, $\wedge$, $\vee$, $\to$, $\approx^\tau$, $\not\approx^\tau$ are interpreted as the corresponding logical operations. Note that R_s never contains any rules rewriting s because s is smaller than any clause containing s. So s can be reducible w.r.t. R_s only when one of its proper subterms is reducible. Since every term has a normal form, adding rules only for the irreducible terms is sufficient.
- By Lemma 4.5, we have $\mathcal{F}(\mathcal{J}(u)\,\mathsf{diff}^{\tau,\upsilon}_{s,t}) = \mathcal{F}(u\,\mathsf{diff}\langle\tau,\upsilon\rangle(s,t))$ for all $u, s, t \in \mathcal{T}_{\mathrm{ground}}(\Sigma_\mathrm{H})$. Since $\mathcal{J}$ is a bijection on ground terms, Lemma 4.36 proves the extensionality condition in Section 2.6.
- The argument congruence condition in Section 2.6 follows from Lemma 4.35 in the same way. □

We employ a variant of Bachmair and Ganzinger's framework of reducing counterexamples [2, Sect. 4.2]. Let $N \subseteq \mathcal{C}_\mathrm{F}$ with $\bot \notin N$. A clause $C_0 \in \mathcal{C}_\mathrm{F}$ is called a *counterexample* if $R_N \not\models_{\mathrm{fol}} C_0$. An inference *reduces* a counterexample C_0 if its main premise is C_0, its side premises are in N and true in R_N, and its conclusion is a counterexample smaller than C_0. An inference system has the *reduction property for counterexamples* if for all $N \subseteq \mathcal{C}_\mathrm{F}$ and all counterexamples $C_0 \in N$, there exists an inference from N that reduces C_0.

Lemma 4.38. *Let $C \in N$ be a counterexample. Let L be a literal in C that is eligible and negative or strictly eligible and positive. We assume that the larger side of L is reducible by a rule $s \to s' \in R_C$. Then the inference system FInf reduces the counterexample C.*

Proof. Let p be the position of C that is located at the larger side of L and reducible by $s \to s'$. We make a case distinction on which case of Definition 4.30 the rule $s \to s'$ originates from:

- (Δ1) Then the root of s is a logical symbol and $s \notin \{\mathsf{T}, \bot\}$. By Lemma 4.32, R_N reduces s to T or to $\bot$.
  - First consider the case where the position p in C is in a literal of the form $s \approx \mathsf{T}$ or $s \approx \bot$. Then FCClausify is applicable to C and the conclusion of this inference is smaller than it. Moreover, the conclusion is equivalent to C by Lemma 4.37.
  - Otherwise, we apply either PFBoolHoist (if s reduces to $\bot$) or PFLoobHoist (if s reduces to T). In both cases, the conclusion of the inference is smaller than C. Moreover, the conclusion is equivalent to C by Lemma 4.37.

– (Δ2) Then R_N reduces s to $\bot$ and $s \notin \{\top, \bot\}$. Due to the presence of the rule $s \to \bot$ in R_C, C must be larger than $s \approx \bot$. So, since p is eligible in C, this position cannot be in a literal of the form $s \approx \top$. It cannot be in a literal of the form $s \approx \bot$ either because $s \approx \bot$ is true in R_N. So we can apply FBoolHoist to reduce the counterexample, again using Lemma 4.37.
– (Δ3) Then s is functional and reducible w.r.t. R_N. Consider the normal form $s \downarrow_{R_N}$ of s w.r.t. R_N. Let $u = \mathcal{F}^{-1}(s)$ and $w = \mathcal{F}^{-1}(s \downarrow_{R_N})$. Then $\mathcal{F}(u) \leftrightarrow^*_{R_N} \mathcal{F}(w)$. By Lemma 4.35, $\mathcal{F}(u\ \mathsf{diff}^{\tau,\upsilon}_{u,w}) \leftrightarrow^*_{R_N} \mathcal{F}(w\ \mathsf{diff}^{\tau,\upsilon}_{u,w})$. Since R_N is oriented by $\succ$, we have $\mathcal{F}(u) = s \succ s \downarrow_{R_N} = \mathcal{F}(w)$. Thus, we can apply FExt to reduce the counterexample, using Remark 4.7. Given the above properties of R_N, the conclusion of this inference is equivalent to the premise. It is also smaller than the premise by $(\mathrm{O5})_\mathrm{F}$ and because $\mathcal{F}(u) \succ \mathcal{F}(w)$.
– (Δ4) Then some clause $D \vee s \approx s'$ smaller than C produces the rule $s \to s'$. We claim that the counterexample C is reduced by the inference

$$\frac{D \vee s \approx s' \quad C[s]}{D \vee C[s']}\,\text{FSup}$$

This superposition is a valid inference:
– s is nonfunctional by (CC2).
– We have $s \succ s'$ by (CC5).
– $D \vee s \approx s' \prec C[s]$ because $D \vee s \approx s'$ produces a rule in R_C.
– The position p of s in C is eligible by assumption of this lemma.
– The literal $s \approx s'$ is eligible in $D \vee s \approx s'$ by (CC6) and (CC7). It is strictly eligible because if $s \approx s'$ also occurred as a literal in D, we would have $R_{D \vee s \approx s'} \cup \{s \to s'\} \models_\mathrm{fol} D$, in contradiction to (CC10).
– If s' is Boolean, then $s' = \top$ by (CC4).

As $D \vee s \approx s'$ is productive, $R_N \not\models_\mathrm{fol} D$ by Lemma 4.34. Hence $D \vee C\,[s']$ is equivalent to $C\,[s']$, which is equivalent to $C\,[s]$ w.r.t. R_N. It remains to show that the new counterexample $D \vee C\,[s']$ is strictly smaller than C. Using $(\mathrm{O2})_\mathrm{F}$, $C[s'] \prec C$ because $s' \prec s$ and $D \prec C$ because $D \vee s \approx s' \prec C$. Thus, the inference reduces the counterexample C. □

Lemma 4.39. *The inference system FInf has the reduction property for counterexamples.*

Proof. Let $C \in N$ be a counterexample—i.e., a clause that is false in R_N. We must show that there is an inference from N that reduces C; i.e., the inference has main premise C, side premises in N that are true in R_N, and a conclusion that is a smaller counterexample than C.

Let L be an eligible literal in C. We proceed by a case distinction:

Case 1: L is of the form $s \not\approx s'$.

– Case 1.1: $s = s'$. Then FEqRes reduces C.
– Case 1.2: $s \neq s'$. Without loss of generality, $s \succ s'$. Since $R_N \not\models_\mathrm{fol} C$, we have $R_C \not\models_\mathrm{fol} C$ by Lemma 4.33. Therefore, $R_C \not\models_\mathrm{fol} s \not\approx s'$ and $R_C \models_\mathrm{fol} s \approx s'$. Thus, s must be reducible by R_C because $s \succ s'$. Therefore, we can apply Lemma 4.38.

Case 2: L is of the form $s \approx s'$. Since $R_N \not\models_\mathrm{fol} C$, we can assume without loss of generality that $s \succ s'$.

– Case 2.1: L is eligible, but not strictly eligible. Then L occurs more than once in C. So we can apply FEqFact to reduce the counterexample.

– Case 2.2: L is strictly eligible and s is reducible by R_C. Then we apply Lemma 4.38.
– Case 2.3: L is strictly eligible and $s = \bot$. Then, since $s \succ s'$, we have $s' = \top$ by $(\mathrm{O4})_{\mathrm{F}}$. So, PFFalseElim reduces the counterexample.
– Case 2.4: L is strictly eligible and s is functional. Then we apply FArgCong to reduce the counterexample. The conclusion is smaller than the premise by $(\mathrm{O5})_{\mathrm{F}}$. By Lemma 4.36, there must be at least one choice of u and w in the FArgCong rule such that the conclusion is a counterexample.
– Case 2.5: L is strictly eligible and $s \neq \bot$ is nonfunctional and not reducible by R_C. Since $R_N \not\models_{\mathrm{fol}} C$, C cannot be productive. So at least one of the conditions of (Δ4) of Definition 4.30 is violated. (CC1), (CC2), (CC5), (CC8), and (CC9) are clearly satisfied. For (CC3), (CC4), (CC6), and (CC7), we argue as follows:
  – (CC3): If s were headed by a logical symbol, then one of the cases of (Δ1) applies. The condition in (Δ1) that any Boolean arguments of s must be $\top$ or $\bot$ is fulfilled by Lemma 4.32 and the fact that the rules applicable to subterms of s in R_N are already contained in R_s. So (Δ1) adds a rewrite rule for s to R_C, contradicting irreducibility of s.
  – (CC4): If s' were a Boolean other than $\top$, since $s \succ s'$, we would have $s \neq \top, \bot$ by $(\mathrm{O4})_{\mathrm{F}}$. Moreover, $s' \succeq \bot$, and thus $C \succeq s \approx \bot$. Since s is not reducible by R_C, is is also irreducible by $R_{s \approx \bot} \subseteq R_C$. So ($\Delta$2) triggers and sets $\Delta_{s \approx \bot} = \{s \to \bot\}$. Since s is not reducible by R_C, we must have $C = s \approx \bot$. But then C istrue in R_N, a contradiction.
  – (CC6): By (CC4), L cannot be selected and thus eligibility implies maximality.
  – (CC7): By (CC4), L cannot be selected. If another literal was selected, L_0 would not be eligible.

  So (CC10) must be violated. Then $R_C \cup \{s \to s'\} \models_{\mathrm{fol}} C'$, where C' is the subclause of C with L removed. However, $R_C \not\models_{\mathrm{fol}} C$, and therefore, $R_C \not\models_{\mathrm{fol}} C'$. Thus, we must have $C' = C'' \vee r \approx t$ for some terms r and t, where $R_C \cup \{s \to s'\} \models_{\mathrm{fol}} r \approx t$ and $R_C \not\models_{\mathrm{fol}} r \approx t$. So $r \neq t$ and without loss of generality we assume $r \succ t$. Moreover $s \to s'$ must participate in the normalization of r or t by $R_C \cup \{s \to s'\}$. Since $s \approx s'$ is maximal in C by (CC6), $r \preceq s$. So the rule $s \to s'$ can be used only as the first step in the normalization of r. Hence $r = s$ and $R_C \models_{\mathrm{fol}} s' \approx t$. Then FEqFact reduces the counterexample. □

Using Lemma 4.39 and the same ideas as for Theorem 4.9 of Bachmair and Ganzinger's framework [2], we obtain the following theorem:

Theorem 4.40. *Let N be a set of closures that is saturated up to redundancy w.r.t. FInf and $\mathit{FRed}_{\mathrm{I}}$, and $\bot \notin N$. Then $R_N \models_{\mathrm{o}\lambda} N$.*

Proof. By Lemma 4.37, it suffices to show that $R_N \models_{\mathrm{fol}} N$. For a proof by contradiction, we assume that $R_N \not\models_{\mathrm{fol}} N$. Then N contains a minimal counterexample, i.e., a clause C with $R_N \not\models_{\mathrm{fol}} C$. Since *FInf* has the reduction property for counterexamples by Lemma 4.39, there exists an inference that reduces C—i.e., an inference ι with main premise C, side premises in N that are true in R_N, and a conclusion $\mathit{concl}(\iota)$ that is smaller than C and false in R_N. By saturation up to redundancy, $\iota \in \mathit{FRed}_{\mathrm{I}}$. By Definition 4.11, we have $\{E \in N \mid E \prec C\} \models_{\mathrm{o}\lambda} \mathit{concl}(\iota)$. By minimality of the counterexample C, the clauses $\{E \in N \mid E \prec C\}$ must be true in R_N, and it follows that $\mathit{concl}(\iota)$ is true in R_N, a contradiction. □

4.4.2. *Indexed Ground Higher-Order Level.*

In this subsubsection, let $\succ$ be an admissible term order for *IGInf* (Definition 4.8), let *igsel* be a selection function on $\mathcal{C}_{\mathrm{IG}}$, and let $N \subseteq \mathcal{C}_{\mathrm{IG}}$ such that N is saturated up to redundancy w.r.t. *IGInf* and $\bot \notin N$. We write R for the term rewrite system $R_{\mathcal{F}(N)}$ constructed in the previous subsubsection w.r.t. $\succ_{\mathcal{F}}$ and $\mathcal{F}(\mathit{igsel})$. We write $t \sim s$ for $\mathcal{F}(t) \leftrightarrow_R^* \mathcal{F}(s)$, where $t, s \in \mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$.

Our goal in this subsubsection is to use R to define a higher-order interpretation that is a model of N. To obtain a valid higher-order interpretation, we need to show that $s\theta \sim s\theta'$ whenever $x\theta \sim x\theta'$ for all x in s.

Lemma 4.41 (Argument congruence)**.** *Let $s \sim s'$ for $s, s' \in \mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$. Let $u \in \mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$. Then $s\ u \sim s'\ u$.*

Proof. Consider the following inference ι:

$$\frac{}{s\ \mathsf{diff}^{\tau,\upsilon}_{s,s'} \not\approx s'\ \mathsf{diff}^{\tau,\upsilon}_{s,s'} \vee s\ u \approx s'\ u}\ \textsc{IGDiff}$$

Since N is saturated, ι is redundant and thus $\mathcal{F}(N) \models \mathcal{F}(\mathit{concl}(\iota))$. Hence $R \models \mathcal{F}(\mathit{concl}(\iota))$ by Theorem 4.40 and Lemma 4.16.

By Lemma 4.35, $R \models \mathcal{F}(s\ \mathsf{diff}^{\tau,\upsilon}_{s,s'}) \approx \mathcal{F}(s'\ \mathsf{diff}^{\tau,\upsilon}_{s,s'})$. It follows that $R \models \mathcal{F}(s\ u \approx s'\ u)$. Thus, $s\ u \sim s'\ u$. □

The following lemma and its proof are essentially identical to Lemma 54 of Bentkamp et al. [5]. We have adapted the proof to use De Bruijn indices, and we have removed the notion of term-ground and replaced it by preprocessing term variables, which arguably would have been more elegant in the original proof as well.

Lemma 4.42. *Let $s \in \mathcal{T}(\Sigma_{\mathrm{I}})$, and let θ, θ' be grounding substitutions such that $x\theta \sim x\theta'$ for all variables x and $\alpha\theta = \alpha\theta'$ for all type variables α. Then $s\theta \sim s\theta'$.*

Proof. In this proof, we work directly on λ-terms. To prove the lemma, it suffices to prove it for any λ-term $s \in \mathcal{T}^{\lambda}(\Sigma_{\mathrm{I}})$. Here, for $t_1, t_2 \in \mathcal{T}^{\lambda}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$, the notation $t_1 \sim t_2$ is to be read as $t_1{\downarrow_\beta} \sim t_2{\downarrow_\beta}$ because $\mathcal{F}$ is defined only on β-normal λ-terms.

Without loss of generality, we may assume that s contains no type variables. If s does contain type variables, we can instead use the term s_0 resulting from instantiating each type variable α in s with $\alpha\theta$. If the result holds for the term s_0, which does not contain type variables, then $s_0\theta \sim s_0\theta'$, and thus the result also holds for s because $s\theta = s_0\theta$ and $s\theta' = s_0\theta'$.

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 $\succ_{\mathsf{CPO}}$ [9] guarantees that

- $\oplus\ t\ s \succ_{\mathsf{CPO}} s$ by applying rule @▷;
- $\oplus\ t\ s \succ_{\mathsf{CPO}} t$ by applying rule @▷ twice;
- $(\lambda\ t)\ s \succ_{\mathsf{CPO}} t\{0 \mapsto s\}$ by applying rule @β.

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 rewriting system, by Kőnig's lemma [16], there exists a maximal number of $\beta\oplus$-reduction steps from each λ-term.

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

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

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 $\big(n_1(s), n_2(s), n_3(s)\big) \in \mathbb{N}^4$, where

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

Case 1: The λ-term s is ground. Then the lemma is trivial.

Case 2: The λ-term s contains $k \geq 2$ variables. Then we can apply the induction hypothesis twice and use the transitivity of $\sim$ as follows. Let x be one of the variables in s. Let $\rho = \{x \mapsto x\theta\}$ the substitution that maps x to $x\theta$ and ignores all other variables. Let $\rho' = \theta'[x \mapsto x]$.

We want to invoke the induction hypothesis on $s\rho$ and $s\rho'$. This is justified because $s\sigma$ $\oplus$-reduces to $s\rho\sigma$ and to $s\rho'\sigma$, for σ as given in the definition of n_1. These $\oplus$-reductions have at least one step because x occurs in s and $k \geq 2$. Hence, $n_1(s) > n_1(s\rho)$ and $n_1(s) > n_1(s\rho')$.

This application of the induction hypothesis gives us $s\rho\theta \sim s\rho\theta'$ and $s\rho'\theta \sim s\rho'\theta'$. Since $s\rho\theta = s\theta$ and $s\rho'\theta' = s\theta'$, this is equivalent to $s\theta \sim s\rho\theta'$ and $s\rho'\theta \sim s\theta'$. Since moreover $s\rho\theta' = s\rho'\theta$, we have $s\theta \sim s\theta'$ by transitivity of $\sim$. The following illustration visualizes the above argument:

$$\begin{array}{ccccc} & s\rho & & & s\rho' \\ \theta\swarrow & & \searrow\theta' & \qquad & \theta\swarrow & & \searrow\theta' \\ s\theta & \underset{\text{IH}}{\sim} & s\rho\theta' & = & s\rho'\theta & \underset{\text{IH}}{\sim} & s\theta' \end{array}$$

Case 3: The λ-term s contains a variable that occurs more than once. Then we rename variable occurrences apart by replacing each occurrence of each variable x by a fresh variable x_i, for which we define $x_i\theta = x\theta$ and $x_i\theta' = x\theta'$. Let s' be the resulting λ-term. Since $s\sigma = s'\sigma$ for σ as given in the definition of n_1, we have $n_1(s) = n_1(s')$. All variables occur only once in s'. Hence, $n_2(s) > 0 = n_2(s')$. Therefore, we can invoke the induction hypothesis on s' to obtain $s'\theta \sim s'\theta'$. Since $s\theta = s'\theta$ and $s\theta' = s'\theta'$, it follows that $s\theta \sim s\theta'$.

Case 4: The λ-term s contains only one variable x, which occurs exactly once.

Case 4.1: The λ-term s is of the form $\mathsf{f}\langle\bar{\tau}\rangle\, \bar{t}$ for some symbol f, some types $\bar{\tau}$, and some λ-terms $\bar{t}$. Then let u be the λ-term in $\bar{t}$ that contains x. We want to apply the induction hypothesis to u, which can be justified as follows. For σ as given in the definition of n_1, consider the longest sequence of $\beta\oplus$-reductions from $u\sigma$. This sequence can be replicated inside $s\sigma = (\mathsf{f}\langle\bar{\tau}\rangle\, \bar{t})\sigma$. Therefore, the longest sequence of $\beta\oplus$-reductions from $s\sigma$ is at least as long—i.e., $n_1(s) \geq n_1(u)$. Since both s and u have only one variable occurrence, we have $n_2(s) = 0 = n_2(u)$. But $n_3(s) > n_3(u)$ because u is a nonground subterm of s.

Applying the induction hypothesis gives us $u\theta \sim u\theta'$. By definition of $\mathcal{F}$, we have $\mathcal{F}((\mathsf{f}\langle\bar{\tau}\rangle\,\bar{t})\theta) = \mathsf{f}^{\bar{\tau}}_m\,\mathcal{F}(\bar{t}\theta)$ and analogously for θ', where m is the length of $\bar{t}$. By congruence of $\approx$ in first-order logic, it follows that $s\theta \sim s\theta'$.

Case 4.2: The λ-term s is of the form $x\,\bar{t}$ for some λ-terms $\bar{t}$. Then we observe that, by assumption, $x\theta \sim x\theta'$. Since x occurs only once, $\bar{t}$ are ground. Then $x\theta\,\bar{t} \sim x\theta'\,\bar{t}$ by applying Lemma 4.41 repeatedly. Hence $s\theta = x\theta\,\bar{t}$ and $s\theta = x\theta'\,\bar{t}$, and it follows that $s\theta \sim s\theta'$.

Case 4.3: The λ-term s is of the form $\lambda\,u$ for some λ-term u. Then we observe that to prove $s\theta \sim s\theta'$, by Lemma 4.36, it suffices to show that $s\theta\;\mathsf{diff}_{s\theta,s\theta'} \sim s\theta'\;\mathsf{diff}_{s\theta,s\theta'}$. Via β-conversion, this is equivalent to $v\theta \sim v\theta'$, where $v = u\{0 \mapsto \mathsf{diff}_{s\theta,s\theta'}\}$. To prove $v\theta \sim v\theta'$, we apply the induction hypothesis on v.

It remains to show that the induction hypothesis applies on v. For σ as given in the definition of n_1, consider the longest sequence of $\beta\oplus$-reductions from $v\sigma$. Since $\mathsf{diff}_{s\theta,s\theta'}$ is not a λ-abstraction, substituting it for 0 will not cause additional $\beta\oplus$-reductions. Hence, the same sequence of $\beta\oplus$-reductions can be applied inside $s\sigma = (\lambda\,u)\sigma$, proving that $n_1(s) \geq n_1(v)$. Since both s and v have only one variable occurrence, $n_2(s) = 0 = n_2(v)$. But $n_3(s) = \mathcal{S}(s) = 1 + \mathcal{S}(u)$ because s is nonground. Moreover, $\mathcal{S}(u) = \mathcal{S}(v) = n_3(v)$. Hence, $n_3(s) > n_3(v)$, which justifies the application of the induction hypothesis.

Case 4.4: The λ-term s is of the form $(\lambda\,u)\,t_0\,\bar{t}$ for some λ-terms u, t_0, and $\bar{t}$. We apply the induction hypothesis on $s' = u\{0 \mapsto t_0\}\,\bar{t}$, justified as follows. For σ as given in the definition of n_1, consider the longest sequence of $\beta\oplus$-reductions from $s'\sigma$. Prepending the reduction $s\sigma \to_\beta s'\sigma$ to it gives us a longer sequence from $s\sigma$. Hence, $n_1(s) > n_1(s')$. The induction hypothesis gives us $s'\theta \sim s'\theta'$. Since $\sim$ is invariant under β-reductions, it follows that $s\theta \sim s\theta'$. □

Using the term rewrite system R, we define a higher-order interpretation $\mathcal{I}^{\mathrm{IG}} = (\mathcal{U}^{\mathrm{IG}}, \mathcal{J}^{\mathrm{IG}}_{\mathrm{ty}}, \mathcal{J}^{\mathrm{IG}}, \mathcal{L}^{\mathrm{IG}})$. The construction proceeds as in the completeness proof of the original λ-superposition calculus [5]. Let $(\mathcal{U}, \mathcal{J}) = R$; i.e., $\mathcal{U}_\tau$ is the universe for the first-order type τ, and $\mathcal{J}$ is the interpretation function. Since the higher-order and first-order type signatures are identical, we can identify ground higher-order and first-order types. We will define a domain $\mathcal{D}_\tau$ for each ground type τ and then let $\mathcal{U}^{\mathrm{IG}}$ be the set of all these domains $\mathcal{D}_\tau$. We cannot identify the domains $\mathcal{D}_\tau$ with the first-order domains $\mathcal{U}_\tau$ because domains $\mathcal{D}_\tau$ for functional types τ must contain functions. Instead, we will define suitable domains $\mathcal{D}_\tau$ and a bijection $\mathcal{E}_\tau$ between $\mathcal{U}_\tau$ and $\mathcal{D}_\tau$ for each ground type τ.

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

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

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

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

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

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 terms $t \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$ and $v \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$ such that $[\![\mathcal{F}(s)]\!]_R = [\![\mathcal{F}(t)]\!]_R$ and $\mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R) = \mathcal{E}_\tau([\![\mathcal{F}(v)]\!]_R)$. Since $\mathcal{E}_\tau$ is bijective, we have $[\![\mathcal{F}(u)]\!]_R = [\![\mathcal{F}(v)]\!]_R$—i.e., $u \sim v$. The terms s, t, u, v are in $\mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$, allowing us to apply Lemma 4.42 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 $s, t \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$ such that for all $u \in \mathcal{T}_{\text{ground}}(\Sigma_{\text{I}})$, we have $[\![\mathcal{F}(s\,u)]\!]_R = [\![\mathcal{F}(t\,u)]\!]_R$. By Lemma 4.36, 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 $\varphi(\mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R)) = \mathcal{E}_\upsilon([\![\mathcal{F}(s\,u)]\!]_R)$ 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.

We define the higher-order universe as $\mathcal{U}^{\text{IG}} = \{\mathcal{D}_\tau \mid \tau \text{ ground}\}$. In particular, by Lemma 4.37, this implies that $\mathcal{D}_o = \{0, 1\} \in \mathcal{U}^{\text{IG}}$ as needed, where 0 is identified with $[\bot]$ and 1 with $[\mathsf{T}]$. Moreover, we define $\mathcal{J}^{\text{IG}}_{\mathsf{ty}}(\kappa)(\mathcal{D}_{\bar\tau}) = \mathcal{D}_{\kappa(\bar\tau)}$ for all $\kappa \in \Sigma_{\mathsf{ty}}$, completing the type interpretation of $\mathcal{I}^{\text{IG}}_{\mathsf{ty}} = (\mathcal{U}^{\text{IG}}, \mathcal{J}^{\text{IG}}_{\mathsf{ty}})$ and ensuring that $\mathcal{J}^{\text{IG}}_{\mathsf{ty}}(o) = \mathcal{D}_o = \{0, 1\}$.

We define the interpretation function $\mathcal{J}^{\text{IG}}$ for symbols $\mathsf{f} : \Pi\bar\alpha_m.\ \tau$ by $\mathcal{J}^{\text{IG}}(\mathsf{f}, \mathcal{D}_{\bar\upsilon_m}) = \mathcal{E}([\![\mathcal{F}(\mathsf{f}\langle\bar\upsilon_m\rangle)]\!]_R)$.

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

Let $a, b \in \{0, 1\}$, $u_0 = \bot$, and $u_1 = \mathsf{T}$. Then, by Lemma 4.37,

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

(I7) Let $\mathcal{D}_\tau \in \mathcal{U}^{\text{IG}}$ and $a', b' \in \mathcal{D}_\tau$. 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

$$\mathcal{J}^{\text{IG}}(\approx, \mathcal{D}_\tau)(a', b') = \mathcal{E}([\![\mathcal{F}(\approx\langle\tau\rangle)]\!]_R)(\mathcal{E}([\![\mathcal{F}(u)]\!]_R), \mathcal{E}([\![\mathcal{F}(v)]\!]_R)) = \mathcal{E}([\![\mathcal{F}(u \approx\langle\tau\rangle\ v)]\!]_R)$$

which is 1 if $a' = b'$ and 0 otherwise by Lemma 4.37. (I8) Similarly $\mathcal{J}^{\text{IG}}(\not\approx, \mathcal{D}_\tau)(a', b') = 0$ if $a' = b'$ and 1 otherwise. This concludes the proof that $\mathcal{J}^{\text{IG}}$ is an interpretation function.

Finally, we need to define the designation function $\mathcal{L}^{\text{IG}}$, which takes a valuation ξ and a λ-expression as arguments. Given a valuation ξ, we choose a grounding substitution θ such that $\mathcal{D}_{\alpha\theta} = \xi_{\mathsf{ty}}(\alpha)$ and $\mathcal{E}([\![\mathcal{F}(x\theta)]\!]_R) = \xi_{\mathsf{te}}(x)$ 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 $\mathcal{E}^{-1}(\xi_{\mathsf{te}}(x))$ is an element of a first-order universe and R is term-generated, there

exists a ground term s such that $[\![s]\!]^{\xi}_R = \mathcal{E}^{-1}(\xi_{\mathsf{te}}(x))$. Choosing one such s and defining $x\theta = \mathcal{F}^{-1}(s)$ gives us a grounding substitution θ with the desired property.

Let $\mathcal{L}^{\mathrm{IG}}(\xi, \lambda\, t) = \mathcal{E}([\![\mathcal{F}((\lambda\, t)\theta)]\!]_R)$. 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 $\mathcal{D}_{\alpha\theta'} = \xi_{\mathsf{ty}}(\alpha)$ for all α and $\mathcal{E}([\![\mathcal{F}(x\theta')]\!]_R) = \xi_{\mathsf{te}}(x)$ 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 4.42 it follows that $[\![\mathcal{F}((\lambda\, t)\theta)]\!]_R = [\![\mathcal{F}((\lambda\, t)\theta')]\!]_R$, which proves that $\mathcal{L}^{\mathrm{IG}}$ is well defined. This concludes the definition of the interpretation $\mathcal{I}^{\mathrm{IG}} = (\mathcal{U}^{\mathrm{IG}}, \mathcal{J}^{\mathrm{IG}}_{\mathsf{ty}}, \mathcal{J}^{\mathrm{IG}}, \mathcal{L}^{\mathrm{IG}})$. It remains to show that $\mathcal{I}^{\mathrm{IG}}$ is proper.

The higher-order interpretation $\mathcal{I}^{\mathrm{IG}}$ relates to the first-order interpretation R as follows:

Lemma 4.43. *Given a ground λ-term $t \in \mathcal{T}^{\lambda}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$, we have*

$$[\![t]\!]_{\mathcal{I}^{\mathrm{IG}}} = \mathcal{E}([\![\mathcal{F}(t{\downarrow_\beta})]\!]_R)$$

Proof. The proof is adapted from the proof of Lemma 40 in Bentkamp et al. [7]. We proceed by induction on t. If t is of the form $\mathsf{f}\langle\bar{\tau}\rangle$, then

$$\begin{aligned}[\![t]\!]_{\mathcal{I}^{\mathrm{IG}}} &= \mathcal{J}^{\mathrm{IG}}(\mathsf{f}, \mathcal{D}_{\bar{\tau}}) \\ &= \mathcal{E}([\![\mathcal{F}(\mathsf{f}\langle\bar{\tau}\rangle)]\!]_R) = \mathcal{E}([\![\mathcal{F}(t{\downarrow_\beta})]\!]_R)\end{aligned}$$

If t is an application $t = t_1\, t_2$, where t_1 is of type $\tau \to \upsilon$, then

$$\begin{aligned}[\![t_1\, t_2]\!]_{\mathcal{I}^{\mathrm{IG}}} &= [\![t_1]\!]_{\mathcal{I}^{\mathrm{IG}}}([\![t_2]\!]_{\mathcal{I}^{\mathrm{IG}}}) \\ &\overset{\mathrm{IH}}{=} \mathcal{E}_{\tau\to\upsilon}([\![\mathcal{F}(t_1{\downarrow_\beta})]\!]_R)(\mathcal{E}_{\tau}([\![\mathcal{F}(t_2{\downarrow_\beta})]\!]_R)) \\ &\overset{\mathrm{Def}\ \mathcal{E}}{=} \mathcal{E}_{\upsilon}([\![\mathcal{F}((t_1\, t_2){\downarrow_\beta})]\!]_R)\end{aligned}$$

If t is a λ-expression, then

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

where θ is a substitution as required by the definition of $\mathcal{L}^{\mathrm{IG}}$. □

We need to show that the interpretation $\mathcal{I}^{\mathrm{IG}}$ is proper. In the proof, we will need the following lemma, which is very similar to the substitution lemma (Lemma 3.1), but we must prove it here for our particular interpretation $\mathcal{I}^{\mathrm{IG}}$ because we have not shown that $\mathcal{I}^{\mathrm{IG}}$ is proper yet.

Lemma 4.44. *Let ρ be a grounding substitution, t be a λ-term, and ξ be a valuation. Moreover, we define a valuation ξ' by $\xi'_{\mathsf{ty}}(\alpha) = [\![\alpha\rho]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}^{\mathrm{IG}}_{\mathsf{ty}}}$ for all type variables α and $\xi'_{\mathsf{te}}(x) = [\![x\rho]\!]^{\xi}_{\mathcal{I}^{\mathrm{IG}}}$ for all term variables x. We then have*

$$[\![t\rho]\!]^{\xi}_{\mathcal{I}^{\mathrm{IG}}} = [\![t]\!]^{\xi'}_{\mathcal{I}^{\mathrm{IG}}}$$

Proof. The proof is adapted from the proof of Lemma 41 in Bentkamp et al. [7]. We proceed by induction on the structure of τ and t. The proof is identical to that of Lemma 3.1, except for the last case, which uses properness of the interpretation, a property we cannot assume

here. However, here, we have the assumption that ρ is a grounding substitution. Therefore, if t is a λ-expression, we argue as follows:

$$\begin{aligned}
[\![(\lambda\, u)\rho]\!]^{\xi}_{\mathfrak{I}^{\mathrm{IG}}} &= [\![\lambda\, u\rho]\!]^{\xi}_{\mathfrak{I}^{\mathrm{IG}}} \\
&= \mathcal{L}^{\mathrm{IG}}(\xi, \lambda\, u\rho) && \text{by the definition of the term denotation} \\
&= \mathcal{E}([\![\mathcal{F}((\lambda\, u)\rho\theta{\downarrow_\beta})]\!]_R) && \text{for some } \theta \text{ by the definition of } \mathcal{L}^{\mathrm{IG}} \\
&= \mathcal{E}([\![\mathcal{F}((\lambda\, u)\rho{\downarrow_\beta})]\!]_R) && \text{because } (\lambda\, u)\rho \text{ is ground} \\
&\overset{*}{=} \mathcal{L}^{\mathrm{IG}}(\xi', \lambda\, u) && \text{by the definition of } \mathcal{L}^{\mathrm{IG}} \text{ and Lemma 4.43} \\
&= [\![\lambda\, u]\!]^{\xi'}_{\mathfrak{I}^{\mathrm{IG}}} && \text{by the definition of the term denotation}
\end{aligned}$$

The step labeled with $*$ is justified as follows: We have $\mathcal{L}^{\mathrm{IG}}(\xi', \lambda\, u) = \mathcal{E}([\![\mathcal{F}((\lambda\, u)\theta'{\downarrow_\beta})]\!]_R)$ by the definition of $\mathcal{L}^{\mathrm{IG}}$, if θ' is a substitution such that $\mathcal{D}_{\alpha\theta'} = \xi'_{\mathsf{ty}}(\alpha)$ for all α and $\mathcal{E}([\![\mathcal{F}(x\theta'{\downarrow_\beta})]\!]_R) = \xi'_{\mathsf{te}}(x)$ for all x. By the definition of ξ' and by Lemma 4.43, ρ is such a substitution. Hence, $\mathcal{L}^{\mathrm{IG}}(\xi', \lambda\, u) = \mathcal{E}([\![\mathcal{F}((\lambda\, u)\rho{\downarrow_\beta})]\!]_R)$. □

Lemma 4.45. *The interpretation $\mathfrak{I}^{\mathrm{IG}}$ is proper.*

Proof. We need to show that $[\![\lambda\, t]\!]^{(\xi_{\mathsf{ty}},\xi_{\mathsf{te}})}_{\mathfrak{I}^{\mathrm{IG}}}(a) = [\![t\{0 \mapsto x\}]\!]^{(\xi_{\mathsf{ty}},\xi_{\mathsf{te}}[x\mapsto a])}_{\mathfrak{I}^{\mathrm{IG}}}$, where x is a fresh variable.

$$\begin{aligned}
[\![\lambda\, t]\!]^{(\xi_{\mathsf{ty}},\xi_{\mathsf{te}})}_{\mathfrak{I}^{\mathrm{IG}}}(a) &= \mathcal{L}^{\mathrm{IG}}((\xi_{\mathsf{ty}}, \xi_{\mathsf{te}}), \lambda\, t)(a) && \text{by the definition of term denotation} \\
&= \mathcal{E}([\![\mathcal{F}((\lambda\, t)\theta{\downarrow_\beta})]\!]_R)(a) && \text{by the definition of } \mathcal{L}^{\mathrm{IG}} \text{ for some } \theta \\
& && \text{such that } \mathcal{E}([\![\mathcal{F}(z\theta)]\!]_R) = \xi_{\mathsf{te}}(z) \text{ for} \\
& && \text{all } z \text{ and } \mathcal{D}_{\alpha\theta} = \xi_{\mathsf{ty}}(\alpha) \text{ for all } \alpha \\
&= \mathcal{E}([\![\mathcal{F}(((\lambda\, t)\theta\, s){\downarrow_\beta})]\!]_R) && \text{by the definition of } \mathcal{E} \\
& && \text{where } \mathcal{E}([\![\mathcal{F}(s)]\!]_R) = a \\
&= \mathcal{E}([\![\mathcal{F}(t\{0 \mapsto x\}(\theta[x \mapsto s]){\downarrow_\beta})]\!]_R) && \text{by } \beta\text{-reduction} \\
& && \text{where } x \text{ is fresh} \\
&= [\![t\{0 \mapsto x\}(\theta[x \mapsto s])]\!]_{\mathfrak{I}^{\mathrm{IG}}} && \text{by Lemma 4.43} \\
&= [\![t\{0 \mapsto x\}]\!]^{(\xi_{\mathsf{ty}},\xi_{\mathsf{te}}[x\mapsto a])}_{\mathfrak{I}^{\mathrm{IG}}} && \text{by Lemma 4.44}
\end{aligned}$$

□

Lemma 4.46. *$\mathfrak{I}^{\mathrm{IG}}$ is term-generated; i.e., for all $\mathcal{D} \in \mathcal{U}^{\mathrm{IG}}$ and all $a \in \mathcal{D}$, there exists a ground type τ such that $[\![\tau]\!]_{\mathfrak{I}^{\mathrm{IG}}_{\mathsf{ty}}} = \mathcal{D}$ and a ground term t such that $[\![t]\!]_{\mathfrak{I}^{\mathrm{IG}}} = a$.*

Proof. In the construction above, it is clear that there is a one-to-one correspondence between ground types and domains, which yields a suitable ground type τ.

Since R is term-generated, there must be a ground term $s \in \mathcal{T}_{\mathrm{F}}$ such that $[\![s]\!]_R = \mathcal{E}^{-1}(a)$. Let $t = \mathcal{F}^{-1}(s)$. Then, by Lemma 4.43, $[\![t]\!]_{\mathfrak{I}^{\mathrm{IG}}} = \mathcal{E}([\![s]\!]_R) = a$. □

Lemma 4.47. *Given $C \in \mathcal{C}_{\mathrm{IG}}$, we have $\mathfrak{I}^{\mathrm{IG}} \models C$ if and only if $R \models \mathcal{F}(C)$.*

Proof. By Lemma 4.43, we have

$$[\![t]\!]_{\mathfrak{I}^{\mathrm{IG}}} = \mathcal{E}([\![\mathcal{F}(t{\downarrow_\beta})]\!]_R)$$

for any $t \in \mathcal{T}_{\mathrm{ground}}(\Sigma_{\mathrm{I}})$. Since $\mathcal{E}$ is a bijection, it follows that a ground literal $s \mathrel{\dot{\approx}} t$ in a clause $C \in \mathcal{C}_{\mathrm{IG}}$ is true in $\mathfrak{I}^{\mathrm{IG}}$ if and only if $\mathcal{F}(s \mathrel{\dot{\approx}} t)$ is true in R. So any closure $C \in \mathcal{C}_{\mathrm{IG}}$ is true in $\mathfrak{I}^{\mathrm{IG}}$ if and only if $\mathcal{F}(C)$ is true in R. □

Theorem 4.48. *Let $N \subseteq \mathcal{C}_{\mathrm{IG}}$ be saturated up to redundancy w.r.t. $IGRed_{\mathrm{I}}$, and $\bot \notin N$. Then $\mathcal{I}^{\mathrm{IG}} \models N$.*

Proof. By Lemma 4.47, it suffices to show that R is a model of N. We apply Theorem 4.40. Lemma 4.16 shows the condition of saturation up to redundancy. □

4.4.3. *Ground Higher-Order Level.* In this subsubsection, let $\succ$ be an admissible term order for *GInf* (Definition 4.9), and let *gsel* be a selection function on $\mathcal{C}_{\mathrm{G}}$.

It is inconvenient to construct a model of N_0 for the G level because $\mathcal{J}$ converts parameters into subscripts. For example, in the model constructed in the previous subsubsection, it can happen that $\mathsf{a} \approx \mathsf{b}$ holds, but $\mathsf{f}_{\mathsf{a}} \approx \mathsf{f}_{\mathsf{b}}$ does not hold, where a and b are constants and f_{a} and f_{b} are constants originating from a constant f with a parameter. For this reason, our completeness result for the G level only constructs a model of $\mathcal{J}(N) \subseteq \mathcal{C}_{\mathrm{IG}}$ instead of $N \subseteq \mathcal{C}_{\mathrm{G}}$. We will overcome this flaw when we lift the result to the H level where the initial clause set can be assumed not to contain any constants with parameters.

Theorem 4.49. *Let $N \subseteq \mathcal{C}_{\mathrm{G}}$ be saturated up to redundancy w.r.t. $GRed_{\mathrm{I}}$, and $\bot \notin N$. Then $\mathcal{I}^{\mathrm{IG}} \models \mathcal{J}(N)$.*

Proof. This follows from Theorem 4.48 and Lemma 4.23. □

4.4.4. *Full Higher-Order Level.* In this subsubsection, let $\succ$ be an admissible term order (Definition 2.12), which by Lemma 4.24 is also an admissible term order for *GInf*, and let *hsel* be a selection function on $\mathcal{C}_{\mathrm{H}}$ (Definition 2.14).

Definition 4.50. A *derivation* is a finite or infinite sequence of sets $(N_i)_{i\geq 0}$ such that $N_i \setminus N_{i+1} \subseteq HRed_{\mathrm{C}}(N_{i+1})$ for all i. A derivation is called *fair* if all *HInf*-inferences from clauses in $\bigcup_i \bigcap_{j\geq i} N_j$ are contained in $\bigcup_i HRed_{\mathrm{I}}(N_i)$.

Lemma 4.51. *The redundancy criteria $HRed_{\mathrm{C}}$ and $HRed_{\mathrm{I}}$ fulfill the following properties, as stated by Waldmann et al. [20]:*

(R2) *if $N \subseteq N'$, then $HRed_{\mathrm{C}}(N) \subseteq HRed_{\mathrm{C}}(N')$ and $HRed_{\mathrm{I}}(N) \subseteq HRed_{\mathrm{I}}(N')$;*
(R3) *if $N' \subseteq HRed_{\mathrm{C}}(N)$, then $HRed_{\mathrm{C}}(N) \subseteq HRed_{\mathrm{C}}(N \setminus N')$ and $HRed_{\mathrm{I}}(N) \subseteq HRed_{\mathrm{I}}(N \setminus N')$;*
(R4) *if $\iota \in HInf$ and $concl(\iota) \in N$, then $\iota \in HRedI(N)$.*

Proof. (R2): This is obvious by definition of clause and inference redundancy.

(R3) for clauses:

Define $\blacktriangleright$ as a relation on sets of pairs of a clause $C \in \mathcal{C}_{\mathrm{H}}$ and a grounding substitution θ, written $C \cdot \theta$, as

$$C \cdot \theta \blacktriangleright D \cdot \rho \quad \text{iff} \quad C\theta \succ D\rho \text{ or } (C\theta = D\rho \text{ and } C \sqsupset D)$$

Clearly, for all $C \in \mathcal{C}_{\mathrm{H}}$ and all $N \subseteq \mathcal{C}_{\mathrm{H}}$, we have $C \in HRed_{\mathrm{C}}(N)$ if and only if for all grounding substitutions θ, we have

$$\{\mathcal{F}(E\zeta) \mid E \in N,\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \models_{\mathrm{o}\lambda} \mathcal{F}(C\theta)$$

Now we are ready to prove (R3). Let $C \in HRed_{\mathrm{C}}(N)$. We must show that $C \in HRed_{\mathrm{C}}(N \setminus N')$. Let θ be a grounding substitution. We must show that

$$\{\mathcal{F}(E\zeta) \mid E \in N \setminus N',\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \models_{\mathrm{o}\lambda} \mathcal{F}(C\theta)$$

Since $C \in \mathit{HRed}_{\mathrm{C}}(N)$, we know that

$$\{\mathcal{F}(E\zeta) \mid E \in N,\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \models_{\mathrm{o}\lambda} \mathcal{F}(C\theta)$$

So it suffices to show that

$$\begin{aligned} &\{\mathcal{F}(E\zeta) \mid E \in N \setminus N',\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \\ \models_{\mathrm{o}\lambda}\ &\{\mathcal{F}(E\zeta) \mid E \in N,\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \end{aligned}$$

Let $E_0 \in N$ and ζ_0 grounding with $E_0 \cdot \zeta_0 \blacktriangleleft C \cdot \theta$. We will show by well-founded induction on $E_0 \cdot \zeta_0$ w.r.t. $\blacktriangleleft$ that

$$\{\mathcal{F}(E\zeta) \mid E \in N \setminus N',\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \models_{\mathrm{o}\lambda} \mathcal{F}(E_0\zeta_0) \qquad (*)$$

Our induction hypothesis states:

$$\begin{aligned} &\{\mathcal{F}(E\zeta) \mid E \in N \setminus N',\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \\ \models_{\mathrm{o}\lambda}\ &\{\mathcal{F}(E\zeta) \mid E \in N,\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft E_0 \cdot \zeta_0\} \end{aligned}$$

If $E_0 \in N \setminus N'$, the claim $(*)$ is obvious. So we may assume that $E_0 \in N'$. The assumption of (R3) states $N' \subseteq \mathit{HRed}_{\mathrm{C}}(N)$, and thus we have

$$\{\mathcal{F}(E\zeta) \mid E \in N,\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft E_0 \cdot \zeta_0\} \models_{\mathrm{o}\lambda} \mathcal{F}(E_0\zeta_0)$$

By the induction hypothesis, this implies $(*)$.

(R3) for inferences:

Inspecting this definition of $\mathit{HRed}_{\mathrm{I}}$ (Definition 2.25), we observe that to show that $\mathit{HRed}_{\mathrm{I}}(N) \subseteq \mathit{HRed}_{\mathrm{I}}(N \setminus N')$, it suffices to prove that

$$\begin{gathered} \{E \in \mathcal{F}(\mathcal{G}(N \setminus N')) \mid E \prec_{\mathcal{F}} \mathcal{F}(C_m\theta_m)\} \\ \models_{\mathrm{o}\lambda} \\ \{E \in \mathcal{F}(\mathcal{G}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(C_m\theta_m)\} \end{gathered}$$

(possibly without the condition $E \prec_{\mathcal{F}} \mathcal{F}(C_m\theta_m)$ for DIFF inferences), where C_m and θ_m are given in the definition of $\mathit{HRed}_{\mathrm{I}}$. We can equivalently write this as

$$\begin{aligned} &\{\mathcal{F}(E\zeta) \mid E \in N \setminus N',\ \zeta \text{ grounding, and } E\zeta \prec C_m\theta_m\} \\ \models_{\mathrm{o}\lambda}\ &\{\mathcal{F}(E\zeta) \mid E \in N,\ \zeta \text{ grounding, and } E\zeta \prec C_m\theta_m\} \end{aligned}$$

Let $E_0 \in N$ and ζ_0 grounding with $E_0\zeta_0 \prec C_m\theta_m$. We must show that

$$\{\mathcal{F}(E\zeta) \mid E \in N \setminus N',\ \zeta \text{ grounding, and } E\zeta \prec C_m\theta_m\} \models_{\mathrm{o}\lambda} \mathcal{F}(E_0\zeta_0) \qquad (\dagger)$$

If $E_0 \in N \setminus N'$, the claim $(\dagger)$ is obvious. So we may assume that $E_0 \in N'$. The assumption of (R3) states $N' \subseteq \mathit{HRed}_{\mathrm{C}}(N)$, and thus $N' \subseteq \mathit{HRed}_{\mathrm{C}}(N \setminus N')$ by (R3) for clauses. So we have

$$\{\mathcal{F}(E\zeta) \mid E \in N \setminus N',\ \zeta \text{ grounding, and } E \cdot \zeta \blacktriangleleft E_0 \cdot \zeta_0\} \models_{\mathrm{o}\lambda} \mathcal{F}(E_0\zeta_0)$$

This implies $(\dagger)$ because for any $E \cdot \zeta$ with $E \cdot \zeta \blacktriangleleft E_0 \cdot \zeta_0$, we have $E\zeta \preceq E_0\zeta_0 \prec C_m\theta_m$.

(R4) Let $\iota \in \mathit{HInf}$ with $\mathit{concl}(\iota) \in N$. We must show that $\iota \in \mathit{HRed}_{\mathrm{I}}(N)$. Let $C_1, \ldots, C_m$ be ι's premises and C_{m+1} its conclusion. Let $\theta_1, \ldots, \theta_{m+1}$ be a tuple of substitutions for which ι is rooted in FInf (Definition 2.24). According to the definition of $\mathit{HRed}_{\mathrm{I}}$ (Definition 2.25), we must show that

– $\mathcal{F}(\mathcal{G}(N)) \models_{\mathrm{o}\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$ if ι is a DIFF inference; and
– $\{E \in \mathcal{F}(\mathcal{G}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(C_m\theta_m)\} \models_{\mathrm{o}\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$ if ι is some other inference.

Since $\mathit{concl}(\iota) \in N$ and $\mathit{concl}(\iota) = C_{m+1}$, we have $C_{m+1} \in N$. Thus, $\mathcal{F}(C_{m+1}\theta_{m+1}) \in \mathcal{F}(\mathcal{G}(N))$. This completes the proof for DIFF inferences because $\mathcal{F}(C_{m+1}\theta_{m+1}) \models_{o\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$. For the other inferences, it remains to prove that $\mathcal{F}(C_{m+1}\theta_{m+1}) \prec_{\mathcal{F}} \mathcal{F}(C_m\theta_m)$.

By Definition 2.24, $\mathcal{F}(C_m\theta_m)$ is the main premise and $\mathcal{F}(C_{m+1}\theta_{m+1})$ is the conclusion of an *FInf* inference. We will show for each *FInf* rule that the conclusion is smaller than the main premise.

By Lemma 4.5, $\succ_{\mathcal{JF}} = \succ_{\mathcal{F}}$. By Lemmas 4.18 and 4.12, it follows that $\succ_{\mathcal{F}}$ is admissible for *FInf*.

For FSUP, we must argue that $C[t] \succ_{\mathcal{F}} D' \vee C[t']$. Since the literal $t \approx t'$ is strictly eligible in D and if t' is Boolean, then $t' = \mathsf{T}$, the literal $t \approx t'$ is strictly maximal in D. Since the position of t is eligible in $C[t]$, it must either occur in a negative literal, in a literal of the form $t \approx \bot$, or in a maximal literal in $C[t]$. If the position of t is in a negative literal or in a literal of the form $t \approx \bot$, then that literal is larger than $t \approx t'$ because if t' is Boolean, then $t' = \mathsf{T}$. Thus, the literal in which t occurs in $C[t]$ is larger than D' because $t \approx t'$ is strictly maximal in D. If the position of t is in a maximal literal of $C[t]$, then that literal is larger than or equal to $t \approx t'$ because $D \prec_{\mathcal{F}} C[t]$, and thus it is larger than D' as well. In $C[t']$, this literal is replaced by a smaller literal because $t \succ_{\mathcal{F}} t'$. So $C[t] \succ_{\mathcal{F}} D' \vee C[t']$.

For FEQRES, clearly, $C' \vee u \not\approx u \succ_{\mathcal{F}} C'$.

For FEQFACT, we have $u \approx v \succeq_{\mathcal{F}} u \approx v'$ and thus $v \succeq_{\mathcal{F}} v'$. Since $u \succ_{\mathcal{F}} v$, we have $u \approx v \succ_{\mathcal{F}} v \not\approx v'$ and thus the premise is larger than the conclusion.

For FCLAUSIFY, it is easy to see that for any of the listed values of s, t, and D, we have $s \approx t \succ_{\mathcal{F}} D$, using $(\mathrm{O3})_{\mathrm{F}}$ and $(\mathrm{O4})_{\mathrm{F}}$. Thus the premise is larger than the conclusion.

For FBOOLHOIST and FLOOBHOIST, we have $u \succ_{\mathcal{F}} \bot$ and $u \succ_{\mathcal{F}} \mathsf{T}$ by $(\mathrm{O4})_{\mathrm{F}}$ because $u \neq \bot$ and $u \neq \mathsf{T}$. Moreover, the occurrence of u in $C[u]$ is required not to be in a literal of the form $u \approx \bot$ or $u \approx \mathsf{T}$, and thus, by $(\mathrm{O4})_{\mathrm{F}}$, it must be in a literal larger than these. It follows that the premise is larger than the conclusion.

For FFALSEELIM, clearly, $C' \vee \bot \approx \mathsf{T} \succ_{\mathcal{F}} C'$.

For FARGCONG, the premise is larger than the conclusion by $(\mathrm{O5})_{\mathrm{F}}$.

For FEXT, we use the condition that $u \succ_{\mathcal{F}} w$ and $(\mathrm{O3})_{\mathrm{F}}$ to show that $C[\mathcal{F}(w)]$ is smaller than the premise. We use $u \succ_{\mathcal{F}} w$ and $(\mathrm{O5})_{\mathrm{F}}$ to show that $\mathcal{F}(u\ \mathsf{diff}\langle\tau,\upsilon\rangle(u,w)) \not\approx \mathcal{F}(w\ \mathsf{diff}\langle\tau,\upsilon\rangle(u,w))$ is smaller than the premise. □

Theorem 4.52. *Given a fair derivation $(N_i)_{i\geq 0}$, where*

1. *N_0 does not have a term-generated model,*
2. *N_0 does not contain parameters,*

we have $\bot \in N_i$ for some index i.

Proof. By Lemma 9 of Waldmann et al. [20], using Lemma 4.51, the limit $N_\infty = \bigcup_i \bigcap_{j\geq i} N_j$ is saturated up to redundancy w.r.t. *HInf* and $\mathit{HRed}_{\mathrm{I}}$. By Lemma 4.29, $\mathcal{G}(N_\infty)$ is saturated up to redundancy w.r.t. *GInf* and $\mathit{GRed}_{\mathrm{I}}$.

For a proof by contradiction, assume that for all i, $\bot \notin N_i$. Then N_∞ does not contain $\bot$ either, and thus $\mathcal{G}(N_\infty)$ does not contain $\bot$. By Lemma 4.49, $\mathcal{I}^{\mathrm{IG}} \models \mathcal{J}(\mathcal{G}(N_\infty))$.

By Lemma 8 of Waldmann et al. [20], using Lemma 4.51, $N_0 \subseteq N_\infty \cup \mathit{HRed}_{\mathrm{C}}(N_\infty)$. Thus, $\mathcal{F}(\mathcal{G}(N_\infty)) \models_{o\lambda} \mathcal{F}(\mathcal{G}(N_0))$. Since $\mathcal{I}^{\mathrm{IG}} \models \mathcal{J}(\mathcal{G}(N_\infty))$, by Lemma 4.47 and Lemma 4.5, it follows that $\mathcal{I}^{\mathrm{IG}} \models \mathcal{J}(\mathcal{G}(N_0))$.

Now $\mathcal{I}^{\mathrm{IG}}$ can be shown to be a model of N_0 as follows. Let $C \in N_0$. Let ξ be a valuation. Since $\mathcal{I}^{\mathrm{IG}}$ is term-generated by Lemma 4.46, there exists a substitution θ such

that $[\![\alpha\theta]\!]_{\mathcal{J}^{\mathrm{IG}}_{\mathsf{ty}}} = \xi_{\mathsf{ty}}(\alpha)$ for all type variables α in C and $[\![x\theta]\!]_{\mathcal{J}^{\mathrm{IG}}} = \xi_{\mathsf{te}}(x)$ for all term variables x in C. Since C does not contain parameters by condition 2 of this theorem, $C\theta \in \mathcal{J}(\mathcal{G}(N_0))$. Thus we have $\mathcal{I}^{\mathrm{IG}} \models C\theta$. By Lemma 3.2, it follows that C is true w.r.t. ξ and $\mathcal{I}^{\mathrm{IG}}$. Since ξ and $C \in N_0$ were arbitrary, we have $\mathcal{I}^{\mathrm{IG}} \models N_0$. This contradicts condition 1 of the present theorem.

□

Lemma 4.53. *Let N be a clause set that does not contain* diff*. If N has a term-generated model, then N has a* diff*-aware model.*

Proof. Let $\mathcal{I} = (\mathcal{I}_{\mathsf{ty}}, \mathcal{J}, \mathcal{L})$ be a model of N. We assume that the signature of $\mathcal{I}$ does not contain diff. We extend it into a diff-aware model $\mathcal{I}' = (\mathcal{I}'_{\mathsf{ty}}, \mathcal{J}', \mathcal{L}')$ as follows.

We define $\mathcal{J}'(\mathsf{diff}, \mathcal{D}_1, \mathcal{D}_2, a, b)$ to be an element $e \in \mathcal{D}_1$ such that $a(e) \neq b(e)$ if such an element exists and an arbitrary element of $\mathcal{D}_1$ otherwise. This ensures that $\mathcal{I}'$ is diff-aware (Definition 1.1).

To define $\mathcal{L}'$, let ξ be a valuation and t be a λ-abstraction. We replace each occurrence of $\mathsf{diff}\langle\tau, \upsilon\rangle(u, w)$ in t with a ground term s that does not contain diff such that $[\![s]\!]_{\mathcal{J}} = \mathcal{J}'(\mathsf{diff}, [\![\tau]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}, [\![\upsilon]\!]^{\xi_{\mathsf{ty}}}_{\mathcal{I}_{\mathsf{ty}}}, [\![u]\!]^{\xi}_{\mathcal{J}}, [\![w]\!]^{\xi}_{\mathcal{J}})$. Such a term s exists because $\mathcal{I}$ is term-generated. We start replacing the innermost occurrences of diff and proceed outward to ensure that the parameters of a replaced diff do not contain diff themselves. Let t' be the result of this replacement. Then we define $\mathcal{L}'(\xi, t) = \mathcal{L}(\xi, t')$. This ensures that $\mathcal{I}'$ is a proper interpretation.

Since N does not contain diff and $\mathcal{I}$ is a model of N, it follows that $\mathcal{I}'$ is a model of N as well. □

Corollary 4.54. *Given a fair derivation $(N_i)_{i\geq 0}$, where*

1. *$N_0 \not\approx \bot$, and*
2. *N_0 does not contain parameters,*

we have $\bot \in N_i$ for some index i.

Proof. By Theorem 4.52 and Lemma 4.53. □

References

[1] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. Log. Comput.*, 4(3):217–247, 1994.

[2] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 19–99. Elsevier and MIT Press, 2001.

[3] Alexander Bentkamp, Jasmin Blanchette, and Matthias Hetzenberger. Term orders for optimistic superposition (unpublished manusscript). `https://nekoka-project.github.io/pubs/optimistic_orders.pdf`.

[4] Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, and Petar Vukmirović. Errata of "Superposition for higher-order logic". `https://matryoshka-project.github.io/pubs/hosup_article_errata.pdf`.

[5] Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, and Petar Vukmirović. Superposition for higher-order logic. *J. Autom. Reason.*, 67(1):10, 2023.

[6] Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, Petar Vukmirovic, and Uwe Waldmann. Errata of "Superposition with lambdas". `https://matryoshka-project.github.io/pubs/lamsup_article_errata.pdf`.

[7] Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, Petar Vukmirovic, and Uwe Waldmann. Superposition with lambdas. *J. Autom. Reason.*, 65(7):893–940, 2021.

[8] Christoph Benzmüller, Nik Sultana, Lawrence C. Paulson, and Frank Theiss. The higher-order prover Leo-II. *J. Autom. Reason.*, 55(4):389–404, 2015.
[9] Frédéric Blanqui, Jean-Pierre Jouannaud, and Albert Rubio. The computability path ordering. *Log. Meth. Comput. Sci.*, 11(4), 2015.
[10] Arthur Charguéraud. The locally nameless representation. *J. Autom. Reason.*, 49(3):363–408, 2012.
[11] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem. *Indag. Math*, 75(5):381–392, 1972.
[12] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, 1979.
[13] Melvin Fitting. *Types, Tableaus, and Gödel's God*. Kluwer, 2002.
[14] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.
[15] Cezary Kaliszyk, Geoff Sutcliffe, and Florian Rabe. TH1: The TPTP typed higher-order form with rank-1 polymorphism. In Pascal Fontaine, Stephan Schulz, and Josef Urban, editors, *PAAR-2016*, volume 1635 of *CEUR Workshop Proceedings*, pages 41–55. CEUR-WS.org, 2016.
[16] Dénes Kőnig. Über eine Schlussweise aus dem Endlichen ins Unendliche. *Acta Sci. Math. (Szeged)*, 3499/2009(3:2–3):121–130, 1927.
[17] Visa Nummelin, Alexander Bentkamp, Sophie Tourret, and Petar Vukmirović. Errata of "Superposition with first-class booleans and inprocessing clausification". `https://matryoshka-project.github.io/pubs/boolsup_errata.pdf`.
[18] Visa Nummelin, Alexander Bentkamp, Sophie Tourret, and Petar Vukmirović. Superposition with first-class Booleans and inprocessing clausification. In André Platzer and Geoff Sutcliffe, editors, *CADE-28*, volume 12699 of *LNCS*, pages 378–395. Springer, 2021.
[19] Stephan Schulz. E - a brainiac theorem prover. *AI Commun.*, 15(2-3):111–126, 2002.
[20] Uwe Waldmann, Sophie Tourret, Simon Robillard, and Jasmin Blanchette. A comprehensive framework for saturation theorem proving. *J. Autom. Reason.*, 66(4):499–539, 2022.