
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 completeness 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 $(\lambda x. t) \approx (\lambda x. \top)$ and $(\lambda x. t) \not\approx (\lambda x. \perp)$. 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, \dots, a_n) . If f is a unary function, we write $f(\bar{a}_n)$ for the elementwise application $(f(a_1), \dots, f(a_n))$.

1.1.1. Types. To define our logic’s types, we fix an infinite set \mathcal{V}_{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 σ of Booleans and a binary type constructor \rightarrow of functions. A *type* is either a type variable $\alpha \in \mathcal{V}_{\text{ty}}$ or an applied type constructor $\kappa(\bar{\tau}_n)$ for some n -ary $\kappa \in \Sigma_{\text{ty}}$ and types $\bar{\tau}_n$. To indicate that an expression e has type τ , we write $e : \tau$.

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 $\Pi\bar{\alpha}_m.\bar{\tau}_n \Rightarrow v$, where $\bar{\tau}_n$ and v are types and $\bar{\alpha}_m$ is a tuple of distinct variables that contains all type variables from $\bar{\tau}_n$ and v . The types $\bar{\tau}_n$ are the types of the parameters of the constant, and v may be a function type if the constant takes nonparameter arguments. We require that Σ contains the logical symbols $\mathbf{T}, \mathbf{L} : o; \neg : o \rightarrow o; \wedge, \vee, \rightarrow : o \rightarrow o \rightarrow o$; and $\approx, \neq : \Pi\alpha. \alpha \rightarrow \alpha \rightarrow o$. 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 $f\langle\bar{v}_m\rangle(\bar{u}_n) : \tau$ for a constant $f \in \Sigma$ with type declaration $\Pi\bar{\alpha}_m.\bar{\tau}_n \Rightarrow \tau$, types \bar{v}_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 $\lambda\langle\tau\rangle t : \tau \rightarrow v$ for a type τ and a λ -preterm $t : v$ such that all De Bruijn indices bound by the new $\lambda\langle\tau\rangle$ have type τ ;
- an application $s t : v$ for λ -preterms $s : \tau \rightarrow v$ 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 $f\langle\bar{v}_m\rangle(\bar{u}_n) : \tau$, 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 $\mathcal{T}^{\lambda\text{pre}}(\Sigma, \mathcal{V})$ 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 $\tau \rightarrow v$ for some types τ and v . It is called *nonfunctional* otherwise.

Given a λ -preterm t and λ -terms s_0, \dots, s_n , we write $t\{0 \mapsto s_0, \dots, n \mapsto s_n\}$ 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, $(f\ 0\ 1\ (\lambda\ g\ 1\ 2))\{0 \mapsto a, 1 \mapsto b\} = f\ a\ b\ (\lambda\ g\ a\ b)$. Given a λ -preterm t and a tuple \bar{s}_n of λ -terms, we abbreviate $t\{0 \mapsto s_1, \dots, (n - 1) \mapsto s_n\}$ as $t\{(0, \dots, n - 1) \mapsto \bar{s}_n\}$.

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 $t = f\langle\bar{\tau}_m\rangle(\bar{u})\ v$ 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 $\mathcal{T}_{\text{ground}}^{\lambda\text{pre}}(\Sigma)$ for the set of ground λ -preterms and $\mathcal{T}_{\text{ground}}^{\lambda}(\Sigma)$ 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} ;
- $f\langle\bar{\tau}\rangle(\bar{u})\bar{t}$ 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 $\alpha \in \mathcal{V}_{\text{ty}}$ 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, substitutions 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 $s \approx t$ stands for either $s \approx t$ or $s \not\approx t$. Nonequational literals are not supported and must be encoded as $s \approx \top$ or $s \approx \perp$. A clause $L_1 \vee \dots \vee L_n$ is a finite multiset of literals. The empty clause is written as \perp . 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* $\mathcal{J}_{\text{ty}} = (\mathcal{U}, \mathcal{J}_{\text{ty}})$ 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}_{ty} associates a function $\mathcal{J}_{\text{ty}}(\kappa) : \mathcal{U}^n \rightarrow \mathcal{U}$ with each n -ary type constructor κ , such that $\mathcal{J}_{\text{ty}}(o) = \{0, 1\}$ and for all domains $\mathcal{D}_1, \mathcal{D}_2 \in \mathcal{U}$, the set $\mathcal{J}_{\text{ty}}(\rightarrow)(\mathcal{D}_1, \mathcal{D}_2)$ is a subset of the function space from \mathcal{D}_1 to \mathcal{D}_2 . The semantics is *standard* if $\mathcal{J}_{\text{ty}}(\rightarrow)(\mathcal{D}_1, \mathcal{D}_2)$ 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}_{ty} and a type valuation ξ_{ty} is recursively defined by $\llbracket \alpha \rrbracket_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}} = \xi_{\text{ty}}(\alpha)$ and $\llbracket \kappa(\bar{\tau}) \rrbracket_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}} = \mathcal{J}_{\text{ty}}(\kappa)(\llbracket \bar{\tau} \rrbracket_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}})$.

Given a type interpretation \mathcal{I}_{ty} and a type valuation ξ_{ty} , a *term valuation* ξ_{te} assigns an element $\xi_{\text{te}}(x) \in \llbracket \tau \rrbracket_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}$ to each variable $x : \tau$. A valuation $\xi = (\xi_{\text{ty}}, \xi_{\text{te}})$ 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}_{ty} associates with each symbol $f : \prod \bar{\alpha}_m. \bar{\tau} \Rightarrow v$, a domain tuple $\bar{\mathcal{D}}_m \in \mathcal{U}^m$, and values $\bar{a} \in \llbracket \bar{\tau} \rrbracket_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}$ a value $\mathcal{J}(f, \bar{\mathcal{D}}_m, \bar{a}) \in \llbracket v \rrbracket_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}$, where ξ_{ty} is a type valuation that maps each α_i to \mathcal{D}_i . We require that

- | | |
|---|---|
| (I1) $\mathcal{J}(\mathbf{T}) = 1$ | (I5) $\mathcal{J}(\neg)(a) = 1 - a$ |
| (I2) $\mathcal{J}(\mathbf{L}) = 0$ | (I6) $\mathcal{J}(\rightarrow)(a, b) = \max\{1 - a, b\}$ |
| (I3) $\mathcal{J}(\mathbf{A})(a, b) = \min\{a, b\}$ | (I7) $\mathcal{J}(\mathbf{A}\mathbf{D})(c, d) = 1$ if $c = d$ and 0 otherwise |
| (I4) $\mathcal{J}(\mathbf{V})(a, b) = \max\{a, b\}$ | (I8) $\mathcal{J}(\mathbf{V}\mathbf{D})(c, d) = 0$ 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}_{ty} is a function that maps a valuation ξ and a λ -expression of type τ to elements of $\llbracket \tau \rrbracket_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}$. 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* $\mathcal{J} = (\mathcal{I}_{\text{ty}}, \mathcal{J}, \mathcal{L})$.

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

An interpretation \mathcal{J} is *proper* if $\llbracket \lambda \langle \tau \rangle t \rrbracket_{\mathcal{J}}^{(\xi_{\text{ty}}, \xi_{\text{te}})}(a) = \llbracket t \{0 \mapsto x\} \rrbracket_{\mathcal{J}}^{(\xi_{\text{ty}}, \xi_{\text{te}}[x \mapsto a])}$ for all λ -expressions $\lambda \langle \tau \rangle t$ and all valuations ξ , where x is a fresh variable. Given an interpretation \mathcal{J} and a valuation ξ , a positive literal $s \approx t$ (resp. negative literal $s \not\approx t$) is true if $\llbracket s \rrbracket_{\mathcal{J}}^\xi$ and $\llbracket t \rrbracket_{\mathcal{J}}^\xi$ 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{J} is a *model* of a set N of clauses, written $\mathcal{J} \models N$, if N is true in \mathcal{J} 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 $\text{diff} : \prod \alpha, \beta. (\alpha \rightarrow \beta, \alpha \rightarrow \beta) \Rightarrow \alpha$, 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{J} *diff-aware* if \mathcal{J} is a model of the extensionality axiom—i.e.,

$$\mathcal{J} \models z \ (\text{diff}(\alpha, \beta)(z, y)) \not\approx y \ (\text{diff}(\alpha, \beta)(z, y)) \vee z \approx y$$

Given two sets M, N of clauses, we write $M \approx N$ 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. \approx but unsound w.r.t. \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 $f a \not\approx b$, a superposition inference at a or $f a$ 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 $\lambda c \not\approx b$ and $f x x \approx c$ can make $\lambda f 0 0 \not\approx b$ redundant (assuming a suitable clause order), but $g a \not\approx b$ and $g \approx f$ cannot make $f a \not\approx b$ 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 $(f\langle\bar{\tau}\rangle(\bar{s})\bar{u}_n)|_{i.p} = v$ for all $f \in \Sigma$, types $\bar{\tau}$, λ -preterms $\bar{s}, \bar{u}_n, v \in \mathcal{T}^{\lambda\text{pre}}(\Sigma)$, and $1 \leq i \leq n$.
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 $1 \leq i \leq n$.
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 $u\ll s \gg_p$ or $u\ll s \gg$ indicates that s is an orange subterm in $u[s]$ at position p , and $u\ll \gg$ 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 $f 0$ and 0 are orange subterms of $\lambda f 0$ 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} ;
- $f\langle\bar{\tau}\rangle(\bar{u}) \bar{t}$ 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 $f\langle\bar{\tau}\rangle(\bar{u}) \bar{t}$, (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 $f\langle\bar{\tau}\rangle(\bar{u}) \bar{t}$, (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 $f\langle\bar{\tau}\rangle(\bar{u}) \bar{t}$ to $\lambda f\langle\bar{\tau}\rangle(\bar{u}) \bar{t} 0$. 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 λt to $\lambda \lambda t 0$, which can be treated analogously to (i).

The other option is an η -reduction of λt 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 $v \bar{w} = t'$. \square

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 $u = f(\bar{t})\bar{s}$ 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 $x : \iota \rightarrow \iota$ be a variable. Let $a : \iota$, $f : \Pi\alpha. \iota \Rightarrow (\iota \rightarrow \iota) \rightarrow \alpha$, and $g : \iota \rightarrow \iota \rightarrow \iota$ be constants. Consider the term $f(\alpha)(a)(\lambda g(x)a)0$. Its green subterms are the entire term (at position ε) and $\lambda g(x)a)0$ (at position 1). Its yellow subterms are the green subterms and $x a$ (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 $g(x)a)0$ (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 $L = s \approx t$ 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 $K = f a \not\approx b$ and $L = c \approx f a$ contains the orange subterm a twice, once at orange position $L.(f a).1$ and once at orange position $K.(f a).1$.

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 $\text{CSU}_X(S)$ or $\text{CSU}(S)$ for an arbitrary complete set of unifiers. Again, we require that all elements of $\text{CSU}(S)$ unify at least the types of the terms pairs in S and that all elements of $\text{CSU}(S)$ 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 \succeq to compare terms that may become equal when instantiated, such as $x b \succeq x a$, where $b \succ a$. 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 $t\ll s' \gg \succ t\ll s \gg$ for ground terms s , s' , and t ;
- (O3) ground yellow subterm property: $t\ll s \gg \succeq s$ for ground terms s and t ;
- (O4) $u \succ \perp \succ \top$ for all ground terms $u \notin \{\top, \perp\}$;
- (O5) $u \succ u \text{ diff}(\tau, v)(s, t)$ for all ground types τ, v and ground terms $s, t, u : \tau \rightarrow v$;
- (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 $t \approx \perp$ 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 $s \doteq t \in C$ 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: λ KBO, inspired by the Knuth–Bendix order, and λ LPO, inspired by the lexicographic path order. Since the companion article defines the orders only on terms, we extend $\succ_{\lambda kbo}$ and $\succ_{\lambda lpo}$ to literals and clauses via the standard extension using multisets [1, Sect. 2.4].

Theorem 2.16. Let $\succ_{\lambda kbo}$ 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 κ assigning argument coefficients to symbols. Assume that these parameters fulfill $w(\mathbf{T}) = w(\perp) = 1$, $w_{db} \geq w(\text{diff})$, $f > \perp > \mathbf{T}$ for all symbols $f \notin \{\mathbf{T}, \perp\}$, and $\kappa(\text{diff}, i) = 1$ for every i . Using the extension to clauses defined above, $\succ_{\lambda kbo}$ 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, $\succ_{g\lambda kbo}$ is the restriction of $\succ_{\lambda kbo}$ to ground terms.

- (O1) By Theorems 3.8 and 3.10 of the companion article, $\succ_{g\lambda kbo}$ 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, $\succ_{g\lambda kbo}$ is compatible with orange contexts and thus also with yellow contexts.
- (O3) By Theorem 3.15 of the companion article, $\succ_{g\lambda kbo}$ enjoys the orange subterm property and thus also the yellow subterm property.
- (O4) By Theorem 3.16 of the companion article, $u \succ_{g\lambda kbo} \perp \succ_{g\lambda kbo} \mathbf{T}$ for all ground terms $u \notin \{\mathbf{T}, \perp\}$, using our assumptions about the weight and precedence of \mathbf{T} and \perp .
- (O5) By Theorem 3.17 of the companion article, $u \succ_{g\lambda kbo} u \text{ diff}(\tau, v)(s, t)$ for all ground types τ, v and ground terms $s, t, u : \tau \rightarrow v$, using our assumptions about the weight and argument coefficients of diff .
- (O6) By definition of our extension of $\succ_{\lambda kbo}$ 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, $\succ_{\lambda kbo}$ is transitive on terms. Since the multiset extension preserves transitivity, it is also transitive on literals.

□

Theorem 2.17. Let $\succ_{\lambda lpo}$ 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 $f > \perp > \mathbf{T}$ for all symbols $f \notin \{\mathbf{T}, \perp\}$, that $\perp \leq ws$, and that $\text{diff} \leq ws$. Using the extension to clauses defined above, $\succ_{\lambda lpo}$ 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, $\succ_{g\lambda lpo}$ is the restriction of $\succ_{\lambda lpo}$ to ground terms.

- (O1) By Theorems 3.21 and 3.22 of the companion article, $\succ_{g\lambda lpo}$ 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, $\succ_{g\lambda lpo}$ is compatible with orange contexts and thus also with yellow contexts.
- (O3) By Theorem 3.25 of the companion article, $\succ_{g\lambda lpo}$ enjoys the orange subterm property and thus also the yellow subterm property.
- (O4) By Theorem 3.26 of the companion article, $u \succ_{g\lambda lpo} \perp \succ_{g\lambda lpo} \mathbf{T}$ for all ground terms $u \notin \{\mathbf{T}, \perp\}$, using our assumptions about the precedence of \mathbf{T} and \perp .
- (O5) By Theorem 3.27 of the companion article, $u \succ_{g\lambda lpo} u \text{ diff}(\tau, v)(s, t)$ for all ground types τ, v and ground terms $s, t, u : \tau \rightarrow v$, using our assumption about the precedence of diff .
- (O6) By definition of our extension of $\succ_{\lambda lpo}$ 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, $\succ_{\lambda\text{po}}$ 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 h_{sel} . We denote this calculus as $HInf^{\succ, h_{sel}}$ 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 $(C_1, C_2, \dots, C_{n+1})$ of clauses, written

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

The clauses C_1, C_2, \dots, 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:

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

1. $\sigma \in CSU(t \equiv u)$;
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. $t\sigma \not\leq t'\sigma$;
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:

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

with the following side conditions, in addition to SUP’s conditions 3 to 7:

1. $\sigma \in CSU(z \ t \equiv u)$;
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:

$$\frac{\overbrace{C' \vee u \not\approx u'}^C}{C'\sigma} \text{EQRES} \quad \frac{\overbrace{C' \vee u' \approx v' \vee u \approx v}^C}{(C' \vee v \not\approx v' \vee u \approx v')\sigma} \text{EQFACT}$$

Side conditions for EQRES:

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

Side conditions for EQFACT:

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

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.

$$\frac{C' \vee s \approx t}{(C' \vee D)\sigma} \text{CLAUSIFY}$$

with the following side conditions:

1. $\sigma \in \text{CSU}(s \equiv s', t \equiv t')$;
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:

$$\begin{array}{lll} (x \wedge y, \top, x \approx \top) & (x \wedge y, \top, y \approx \top) & (x \wedge y, \perp, x \approx \perp \vee y \approx \perp) \\ (x \vee y, \top, x \approx \top \vee y \approx \top) & (x \vee y, \perp, x \approx \perp) & (x \vee y, \perp, y \approx \perp) \\ (x \rightarrow y, \top, x \approx \perp \vee y \approx \top) & (x \rightarrow y, \perp, x \approx \top) & (x \rightarrow y, \perp, y \approx \perp) \\ (x \bowtie \langle \alpha \rangle y, \top, x \approx y) & (x \bowtie \langle \alpha \rangle y, \perp, x \not\approx y) & \\ (x \not\bowtie \langle \alpha \rangle y, \top, x \not\approx y) & (x \not\bowtie \langle \alpha \rangle y, \perp, x \approx y) & \\ (\neg x, \top, x \approx \perp) & (\neg x, \perp, x \approx \top) & \end{array}$$

$$\frac{C \langle u \rangle}{(C \langle \perp \rangle \vee u \approx \top)\sigma} \text{BOOLHOIST} \quad \frac{C \langle u \rangle}{(C \langle \top \rangle \vee u \approx \perp)\sigma} \text{LOOBHOIST}$$

each with the following side conditions:

1. σ is the most general type substitution such that $u\sigma$ is of Boolean type;
2. u is neither \top nor \perp , 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 $u \approx \perp$ or $u \approx \top$.

$$\frac{C\langle u \rangle}{(C\langle z \perp \rangle \vee x \approx \top)\sigma} \text{FLUIDBOOLHOIST}$$

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. $\sigma \in \text{CSU}(z x \equiv u)$;
5. $(z \perp)\sigma \neq (z x)\sigma$;
6. $z\sigma \neq \lambda 0$;
7. $x\sigma \neq \top$ and $x\sigma \neq \perp$;
8. the position of u is eligible in C w.r.t. σ .

$$\frac{C\langle u \rangle}{(C\langle z \top \rangle \vee x \approx \perp)\sigma} \text{FLUIDLOOBHOIST}$$

with the same side conditions as FLUIDBOOLHOIST, but where \perp is replaced by \top in condition 5.

$$\frac{\overbrace{C' \vee s \approx t}^C}{C'\sigma} \text{FALSEELIM}$$

with the following side conditions:

1. $\sigma \in \text{CSU}(s \equiv \perp, t \equiv \top)$;
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.

$$\frac{\overbrace{C' \vee s \approx s'}^C}{C'\sigma \vee s\sigma x \approx s'\sigma x} \text{ARGCONG}$$

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 $\{\alpha \mapsto (\beta \rightarrow \gamma)\}$ 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.

$$\frac{C\langle u \rangle}{C\sigma\langle y \rangle \vee u\sigma (\text{diff}\langle \tau, v \rangle(u\sigma, y)) \not\approx y (\text{diff}\langle \tau, v \rangle(u\sigma, y))} \text{EXT}$$

with the following side conditions:

1. σ is the most general type substitution such that $u\sigma$ is of type $\tau \rightarrow v$ for some τ and v ;
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 .

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

with the following side conditions:

1. u is variable-headed;
2. $u\sigma$ is nonfunctional;
3. x and y are fresh variables of type $\alpha \rightarrow \beta$, and z is a fresh variable of function type from $\alpha \rightarrow \beta$ to the type of u ;
4. $\sigma \in \text{CSU}(S, z x \equiv u)$;
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 .

$$\frac{}{y (\text{diff}\langle \alpha, \beta \rangle(y, z)) \not\approx z (\text{diff}\langle \alpha, \beta \rangle(y, z)) \vee y x \approx z x} \text{DIFF}$$

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

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 $\mathcal{F}(t) = \text{fun}_t$. Otherwise, t is of the form $f\langle \bar{\tau} \rangle(\bar{u}) \bar{t}_m$, and we define $\mathcal{F}(t) = f_{\bar{u}}^{\bar{\tau}}(\mathcal{F}(\bar{t}_1), \dots, \mathcal{F}(\bar{t}_m))$.

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. \square

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

- Interpreted Booleans: The domain of the Boolean type has exactly two elements, $\llbracket \top \rrbracket_J$ and $\llbracket \perp \rrbracket_J$, and the symbols \neg , \wedge , \vee , \rightarrow , \approx^τ , $\not\approx^\tau$ are interpreted as the corresponding logical operations.
- Extensionality w.r.t. diff: For all ground $u, v : \tau \rightarrow v$, if $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u \text{ diff } \langle \tau, v \rangle(s, t)) \approx \mathcal{F}(v \text{ diff } \langle \tau, v \rangle(s, t))$ for all ground $s, t : \tau \rightarrow v$, then $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u) \approx \mathcal{F}(v)$.
- Argument congruence w.r.t. diff: For all ground $u, v, s, t : \tau \rightarrow v$, if $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u) \approx \mathcal{F}(v)$, then $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u \text{ diff } \langle \tau, v \rangle(s, t)) \approx \mathcal{F}(v \text{ diff } \langle \tau, v \rangle(s, t))$.

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 and can be used to justify subsumption. For this second condition, we fix a well-founded partial order \sqsupseteq on \mathcal{C}_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 $C \sqsupseteq D$ 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 $C \in HRed_C(N)$ 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 $C \sqsupseteq D$ 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.

$$\begin{array}{c}
 \frac{\overbrace{D' \vee t \approx t'}^D \quad C[t]}{D' \vee C[t']} \text{FSUP} \quad \frac{\overbrace{C' \vee u \not\approx u}^C}{C'} \text{FEQRES} \\
 \\
 \frac{\overbrace{C' \vee u \approx v' \vee u \approx v}^C}{C' \vee v \not\approx v' \vee u \approx v} \text{FEQFACT} \quad \frac{C' \vee s \approx t}{C' \vee D} \text{FCLAUSIFY} \\
 \\
 \frac{C[u]}{(C[\perp] \vee u \approx \top)} \text{FBOOLHOIST} \quad \frac{C[u]}{C[\top] \vee u \approx \perp} \text{FLOOBHOIST} \\
 \\
 \frac{C'}{\overbrace{C' \vee \perp \approx \top}^C} \text{FFALSEELIM} \\
 \\
 \frac{C'}{C' \vee \mathcal{F}(s) \approx \mathcal{F}(s')} \text{FARGCONG} \\
 \\
 \frac{C[\mathcal{F}(u)]}{C[\mathcal{F}(v)] \vee \mathcal{F}(u \text{ diff } \langle \tau, v \rangle(u, v)) \not\approx \mathcal{F}(v \text{ diff } \langle \tau, v \rangle(u, v))} \text{FEXT} \\
 \\
 \frac{}{\mathcal{F}(u \text{ diff } \langle \tau, v \rangle(u, v)) \not\approx \mathcal{F}(v \text{ diff } \langle \tau, v \rangle(u, v)) \vee \mathcal{F}(u s) \approx \mathcal{F}(v s)} \text{FDIFF}
 \end{array}$$

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:

$$\begin{array}{lll}
 (u \wedge v, \top, u \approx \top) & (u \wedge v, \top, v \approx \top) & (u \wedge v, \perp, u \approx \perp \vee v \approx \perp) \\
 (u \vee v, \top, u \approx \top \vee v \approx \top) & (u \vee v, \perp, u \approx \perp) & (u \vee v, \perp, v \approx \perp) \\
 (u \rightarrow v, \top, u \approx \perp \vee v \approx \top) & (u \rightarrow v, \perp, u \approx \top) & (u \rightarrow v, \perp, v \approx \perp) \\
 (u \approx^\tau v, \top, u \approx v) & (u \approx^\tau v, \perp, u \not\approx v) & \\
 (u \not\approx^\tau v, \top, u \not\approx v) & (u \not\approx^\tau v, \perp, u \approx v) & \\
 (\neg u, \top, u \approx \perp) & (\neg u, \perp, u \approx \top) &
 \end{array}$$

Side conditions for FBOOLHOIST and FLOOBHOIST:

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

Side conditions for FFALSEELIM:

1. $\perp \approx \top$ is strictly eligible in C .

Side conditions for FARGCONG:

1. s is of type $\tau \rightarrow v$;
2. u, v are ground terms of type $\tau \rightarrow v$;
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 $\tau \rightarrow v$;
3. v is a ground term of type $\tau \rightarrow v$;
4. $u \succ v$.

Side conditions for FDIFF:

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

Definition 2.23. We convert a selection function h_{sel} on higher-order clauses into a selection function $\mathcal{F}(h_{sel})$ 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 $\iota \in HInf^{\succ, h_{sel}}$ for a term order \succ and a selection function h_{sel} . Let C_1, \dots, C_m be its premises and C_{m+1} its conclusion. Let $(\theta_1, \dots, \theta_{m+1})$ be a tuple of grounding substitutions. We say that ι is *rooted* in $FInf$ for $(\theta_1, \dots, \theta_{m+1})$ 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 $FInf^{\succ, \mathcal{F}(h_{sel})}$ 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 $N \subseteq C_H$. Let $\iota \in HInf$ an inference with premises C_1, \dots, C_m and conclusion C_{m+1} . We define $HRed_1$ so that $\iota \in HRed_1(N)$ if for all substitutions $(\theta_1, \dots, \theta_{m+1})$ for which ι is rooted in $FInf$ (Definition 2.24), we have

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

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 $\models_{\text{o}\lambda}$ under the encoding \mathcal{F} .

Our analogue of clause subsumption is the following.

$$\frac{C \quad C\sigma \vee D}{C} \text{SUBSUMPTION}$$

with the following side condition:

1. $D \neq \perp$ or $C\sigma \sqsupseteq C$.

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 $\mathcal{F}(C\sigma\theta) \models_{\text{o}\lambda} \mathcal{F}((C\sigma \vee D)\theta)$ and $C\sigma\theta \prec (C\sigma \vee D)\theta$. If $D = \perp$, we apply condition 2 of Definition 2.22, which holds by condition 1 of SUBSUMPTION. \square

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 $u \ll s \gg_p$ w.r.t. $\downarrow_{\beta\eta}$ can be disassembled into a sequence $s_1 \dots s_k$ 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, \dots, 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 $i < k$. \square

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 $u \ll v \gg_p$ and $u \ll v' \gg_p$ 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

$$\begin{aligned} \{\mathcal{F}(v\{(0, \dots, k-1) \mapsto \bar{t}_k\}) \approx v'\{(0, \dots, k-1) \mapsto \bar{t}_k\}) \mid \text{each } t_i \text{ of the form } \text{diff}\langle_, _\rangle(_, _) \} \\ \models_{o\lambda} \mathcal{F}(u\ll v \gg_p \approx u\ll v' \gg_p) \end{aligned}$$

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

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

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

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

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

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

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

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

$$\mathcal{I} \models_{o\lambda} \mathcal{F}(u\ll v \gg_p \approx u\ll v' \gg_p) \quad \square$$

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

$$\frac{t \approx t' \quad C\ll v \gg}{t \approx t' \quad C\ll v' \gg} \text{DEMOD}$$

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. $C\ll v \gg \succ C\ll v' \gg$;
3. for each tuple \bar{t}_k , where each t_i is of the form $\text{diff}\langle_, _\rangle(_, _)$, we have $C\ll v \gg \succ v\{(0, \dots, k-1) \mapsto \bar{t}_k\} \approx v'\{(0, \dots, k-1) \mapsto \bar{t}_k\}$;

Remark 2.29. In general, it is unclear how to compute condition 3 of DEMOD. For λ KBO and λ LPO described in Section 2.4, however, the condition can easily be overapproximated by $C\ll v \gg \succ v \approx v'$, 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 $\text{diff}\langle_, _\rangle(_, _)$. Then $u \succ s'$ (resp. $u \succsim s'$).

PROOF FOR λ KBO: By induction on the rule deriving $u \succ s$ or $u \succsim s$. Since we assume in Section 2.4 that $w_{\text{db}} \geq w(\text{diff})$ and $\kappa(\text{diff}, i) = 1$ 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 λ LPO: By induction on the rule deriving $u \succ s$ or $u \succsim s$. Considering that we assume in Section 2.4 that $ws \geq \text{diff}$, 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 substitution. We apply condition 1 of Definition 2.22, using $C\langle v \rangle$ for C . Let $T = \{\bar{t}_k \mid \text{each } t_i \text{ is a ground term of the form } \text{diff}\langle _, _ \rangle(_, _) \}$ 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,

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

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

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

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

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

Our variant of equality subsumption is the following:

$$\frac{t \approx t' \quad C' \vee s\langle v \rangle \approx s'\langle v' \rangle}{t \approx t'} \text{ EQUALITYSUBSUMPTION}$$

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. for each tuple \bar{t} , where each t_i is of the form $\text{diff}\langle _, _ \rangle(_, _)$, we have $C\langle v \rangle \succ v\{(0, \dots, k-1) \mapsto \bar{t}_k\} \approx v'\{(0, \dots, k-1) \mapsto \bar{t}_k\}$;

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. \square

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 $\tau \rightarrow v$ be the type of $s\theta$ and $s'\theta$. By the extensionality property of $\models_{\text{o}\lambda}$, we have

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

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

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. \square

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. \square

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. \square

The following two rules normalize negative literals with \top and \perp into positive literals.

$$\frac{C' \vee s \not\approx \top}{C' \vee s \approx \perp} \text{ NOTTRUE} \quad \frac{C' \vee s \not\approx \perp}{C' \vee s \approx \top} \text{ NOTFALSE}$$

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. \square

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

$$\frac{\overbrace{C' \vee s \not\approx s'}^C}{C' \vee s \text{ diff}(\tau, v)(s, s') \not\approx s' \text{ diff}(\tau, v)(s, s')} \text{ NEGEXT}$$

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

$$\mathcal{F}((C' \vee s \text{ diff}(\tau, v)(s, s') \not\approx s' \text{ diff}(\tau, v)(s, s'))\theta) \models_{o\lambda} \mathcal{F}((C' \vee s \not\approx s')\theta)$$

By (O5), we have $(C' \vee s' \text{ diff}(\tau, v)(s, s') \not\approx s \text{ diff}(\tau, v)(s, s'))\theta \prec (C' \vee s \not\approx s')\theta$. Thus condition 1 of Definition 2.22 applies. \square

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 $t \approx \perp$. 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) $p a \approx T$
- (2) $q b \approx T$
- (3) $r c \approx T$
- (4) $p x \approx \perp \vee q y \approx \perp \vee r z \approx \perp$

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) $q y \approx \perp \vee r z \approx \perp$
- (6) $p x \approx \perp \vee r z \approx \perp$
- (7) $p x \approx \perp \vee q y \approx \perp$

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: $p x \approx \perp$, $q y \approx \perp$ and $r z \approx \perp$. 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 $r z \approx \perp$ 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 $\iota \rightarrow \iota$.

- (1) $f \approx g$
- (2) $f \not\approx g$

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) $f x \approx g x$ (by ARGCONG from (1))
- (4) $f \text{ diff}(f, g) \not\approx g \text{ diff}(f, g)$ (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) $\text{map}(\lambda u. \text{sqrt}(\text{add } u 1)) x \not\approx \text{map}(\lambda u. \text{sqrt}(\text{add } 1 u)) x$
- (2) $\text{add } u v \approx \text{add } v u$

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

- (3) $\text{sqrt}(\text{add}(\text{diff}_{\lambda u. \text{sqrt}(\text{add } u 1), z}) 1) \not\approx z (\text{diff}_{\lambda u. \text{sqrt}(\text{add } u 1), z}) \vee \text{map } z x \not\approx \text{map}(\lambda u. \text{sqrt}(\text{add } 1 u)) x$ (by EXT from (1))
- (4) $\text{sqrt}(\text{add} \text{diff}_{\lambda u. \text{sqrt}(\text{add } u 1), \lambda u. \text{sqrt}(\text{add } 1 u)} 1) \not\approx \text{sqrt}(\text{add } 1 \text{diff}_{\lambda u. \text{sqrt}(\text{add } u 1), \lambda u. \text{sqrt}(\text{add } 1 u)})$ (by EQRES from (3))
- (5) $\text{sqrt}(\text{add } 1 \text{diff}_{\lambda u. \text{sqrt}(\text{add } u 1), \lambda u. \text{sqrt}(\text{add } 1 u)}) \not\approx \text{sqrt}(\text{add } 1 \text{diff}_{\lambda u. \text{sqrt}(\text{add } u 1), \lambda u. \text{sqrt}(\text{add } 1 u)})$ (by SUP from (2), (4))
- (6) \perp (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

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

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) $(\lambda x. p x) \approx (\lambda x. \top)$
- (2) $p a \approx \perp$

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

- (3) $p x \approx \top$ (by ARGCONG from (1))
- (4) $\top \approx \perp$ (by SUP from (2), (3))
- (5) \perp (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) $(\lambda x. p x) \not\approx (\lambda x. \top)$
- (2) $p x \approx \top$

We can derive a contradiction as follows:

- (3) $p \text{ diff}(\iota, o)(\lambda x. p x, \lambda x. \top) \not\approx \top$ (by NEGEXT from (1))
- (4) $p \text{ diff}(\iota, o)(\lambda x. p x, \lambda x. \top) \approx \perp$ (by NOTTRUE from (3))
- (5) $\top \approx \perp$ (by SUP from (2), (4))
- (6) \perp (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) $b \approx a$
- (2) $(\lambda x. (\neg p x y) \wedge (p x y \vee y \neq a)) \not\approx (\lambda x. \perp)$
- (3) $(\lambda x. (\neg p x b) \wedge (p x b \vee b \neq a)) \not\approx (\lambda x. \perp)$

Note that the clauses $(\lambda x. \dots) \not\approx (\lambda x. \perp)$ can be read as $\exists x. \dots$ 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 $p x b \vee b \neq a$ in (3) is false, and since $(\neg p x b) \wedge (p x b)$ is clearly false, clause (3) is false.

For the following derivation, we assume $b \succ a$. Applying NEGEXT to (2) and (3) followed by NOTFALSE yields

- (4) $\neg p \text{ diff}_{\lambda x. (\neg p x y) \wedge (p x y \vee y \neq a)}, \lambda x. \perp y \wedge p \text{ diff}_{\lambda x. (\neg p x y) \wedge (p x y \vee y \neq a)}, \lambda x. \perp y \vee y \neq a \approx \top$
- (5) $\neg p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \neq a)}, \lambda x. \perp b \wedge p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \neq a)}, \lambda x. \perp b \vee b \neq a \approx \top$

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

- (6) $p \text{ diff}_{\lambda x. (\neg p x y) \wedge (p x y \vee y \neq a)}, \lambda x. \perp y \approx \perp$
- (7) $p \text{ diff}_{\lambda x. (\neg p x y) \wedge (p x y \vee y \neq a)}, \lambda x. \perp y \approx \top \vee y \neq a$
- (8) $p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \neq a)}, \lambda x. \perp b \approx \perp$
- (9) $p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \neq a)}, \lambda x. \perp b \approx \top \vee b \neq a$

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

$$(10) p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \neq a)}, \lambda x. \perp a \approx \top$$

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 \sqsupseteq . The following clauses remain:

- (1) $b \approx a$
- (6) $p \text{ diff}_{\lambda x. (\neg p x y) \wedge (p x y \vee y \neq a)}, \lambda x. \perp y \approx \perp$
- (7) $p \text{ diff}_{\lambda x. (\neg p x y) \wedge (p x y \vee y \neq a)}, \lambda x. \perp y \approx \top \vee y \neq a$
- (10) $p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \neq a)}, \lambda x. \perp a \approx \top$

Assuming that the negative literal in (7) is selected and that $b \succ a$, 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:

$$(11) \quad p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \not\simeq a), \lambda x. \perp} b \approx T \vee a \not\simeq a$$

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 $\mathcal{I} = (\mathcal{I}_{\text{ty}}, \mathcal{J}, \mathcal{L})$ and any valuation ξ ,*

$$\llbracket t\theta \rrbracket_{\mathcal{J}}^{\xi} = \llbracket t \rrbracket_{\mathcal{J}}^{\xi'}$$

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

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

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

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

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

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

CASE $t = s v$:

$$\begin{aligned} \llbracket t\theta \rrbracket_{\mathcal{J}}^{\xi} &= \llbracket s\theta v\theta \rrbracket_{\mathcal{J}}^{\xi} \\ &= \llbracket s\theta \rrbracket_{\mathcal{J}}^{\xi} (\llbracket v\theta \rrbracket_{\mathcal{J}}^{\xi}) \quad (\text{by definition}) \\ &= \llbracket s \rrbracket_{\mathcal{J}}^{\xi'} (\llbracket v \rrbracket_{\mathcal{J}}^{\xi'}) \quad (\text{by induction hypothesis}) \\ &= \llbracket s v \rrbracket_{\mathcal{J}}^{\xi'} \quad (\text{by definition}) \\ &= \llbracket t \rrbracket_{\mathcal{J}}^{\xi'} \end{aligned}$$

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

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

□

Lemma 3.2 (Substitution Lemma for Clauses). *Let θ be a substitution, and let C be a clause. For any proper interpretation $\mathcal{I} = (\mathcal{I}_{\text{ty}}, \mathcal{J}, \mathcal{L})$ 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 $\xi'_{\text{ty}}(\alpha) = \llbracket \alpha\theta \rrbracket_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}$ for type variables α and $\xi'_{\text{te}}(x) = \llbracket x\theta \rrbracket_{\mathcal{J}}^{\xi}$ 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 $\llbracket s\theta \rrbracket_{\mathcal{J}}^{\xi}$ and $\llbracket t\theta \rrbracket_{\mathcal{J}}^{\xi}$ are equal (resp. different). By Lemma 3.1, $\llbracket s\theta \rrbracket_{\mathcal{J}}^{\xi}$ and $\llbracket t\theta \rrbracket_{\mathcal{J}}^{\xi}$ are equal (resp. different) if and only if $\llbracket s \rrbracket_{\mathcal{J}}^{\xi}$ and $\llbracket t \rrbracket_{\mathcal{J}}^{\xi}$ 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}, ξ for all valuations ξ .

For most rules, it suffices to make distinctions on the truth under \mathcal{I}, ξ 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 $\text{diff} : \Pi\alpha, \beta. (\alpha \rightarrow \beta, \alpha \rightarrow \beta) \Rightarrow \alpha$

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

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

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

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

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

Convention 4.2. In the subscripts of constants fun_t in $\mathcal{F}(\Sigma_H)$ and $\mathcal{F}(\Sigma_I)$, we identify each term of the form $f\langle \bar{v}_m \rangle(\bar{t}_n) \in \mathcal{T}_{\text{ground}}(\Sigma_H)$ with the term $f_{\bar{t}_n}^{\bar{v}_m} \in \mathcal{T}_{\text{ground}}(\Sigma_I)$, whenever $n > 0$. Using this identification, we can consider the first-order signatures $\mathcal{F}(\Sigma_H)$ and $\mathcal{F}(\Sigma_I)$ 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}_X for the set of terms and \mathcal{C}_X for the set of clauses of a given level X :

Level	Terms	Clauses
F	ground first-order terms over $\mathcal{F}(\Sigma_I)$	clauses over \mathcal{T}_F
IG	$\mathcal{T}_{\text{ground}}(\Sigma_I)$	clauses over \mathcal{T}_{IG}
G	$\mathcal{T}_{\text{ground}}(\Sigma_H)$	clauses over \mathcal{T}_G
H	$\mathcal{T}(\Sigma_H)$	clauses over \mathcal{T}_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 $\mathcal{T}_F/\mathcal{C}_F$ and as an encoding from $\mathcal{T}_G/\mathcal{C}_G$ to $\mathcal{T}_F/\mathcal{C}_F$. 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}_G to \mathcal{T}_{IG} by encoding any occurrence of a constant with parameters $f\langle \bar{v} \rangle(\bar{u})$ as $f_{\bar{u}}^{\bar{v}}$. Formally:

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

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

Lemma 4.5. Let $t \in \mathcal{T}_{\text{ground}}(\Sigma_H)$, and let C be a clause over $\mathcal{T}_{\text{ground}}(\Sigma_H)$. 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 $\text{fun}_{\mathcal{J}(t)}$ (Convention 4.2). For C , the claim holds because \mathcal{J} and \mathcal{F} map each side of each literal individually. \square

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 $FInf^{\succ, sel}$ 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:

- (O1)_F the relation \succ on ground terms is a well-founded total order;
- (O2)_F ground compatibility with contexts: if $s' \succ s$, then $s'[t] \succ s[t]$;
- (O3)_F ground subterm property: $t[s] \succ s$ for ground terms s and t ;
- (O4)_F $u \succ \perp \succ \top$ for all ground terms $u \notin \{\top, \perp\}$;
- (O5)_F $\mathcal{F}(u) \succ \mathcal{F}(u \text{ diff}_{s,t}^{\tau,v})$ for all $s, t, u : \tau \rightarrow v \in \mathcal{T}_{\text{ground}}(\Sigma_I)$;
- (O6)_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, v from $\mathcal{T}_{\text{ground}}(\Sigma_I)$ instead of $\mathcal{T}_{\text{ground}}(\Sigma_H)$ and replacing $\text{diff}(\tau, v)(u, v)$ with $\text{diff}_{u,v}^{\tau,v}$.

4.2.1. Indexed Ground Higher-Order Level. The calculus $IGInf^{\succ, sel}$ 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 $\mathcal{T}_{\text{ground}}(\Sigma_I)$, and on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_I)$. Such a relation \succ is an *admissible term order for $IGInf$* if it fulfills the following properties:

- (O1)_{IG} the relation \succ on ground terms is a well-founded total order;
- (O2)_{IG} ground compatibility with yellow contexts: $s' \succ s$ implies $t \ll s' \gg \succ t \ll s \gg$ for ground terms s, s' , and t ;
- (O3)_{IG} ground yellow subterm property: $t \ll s \gg \succeq s$ for ground terms s and t ;
- (O4)_{IG} $u \succ \perp \succ \top$ for all ground terms $u \notin \{\top, \perp\}$;
- (O5)_{IG} $u \succ u \text{ diff}_{s,t}^{\tau,v}$ for all ground terms $s, t, u : \tau \rightarrow v$.
- (O6)_{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 $IGInf^{\succ, sel}$ (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 $t' = \top$.

$$\frac{\overbrace{C' \vee u \not\approx u}^C}{C'} \text{IGEQRES}$$

$$\frac{\overbrace{C' \vee u \approx v' \vee u \approx v}^C}{C' \vee v \not\approx v' \vee u \approx v'} \text{IGEQFACT}$$

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:

$(u \wedge v, \top, u \approx \top)$	$(u \wedge v, \top, v \approx \top)$	$(u \wedge v, \perp, u \approx \perp \vee v \approx \perp)$
$(u \vee v, \top, u \approx \top \vee v \approx \top)$	$(u \vee v, \perp, u \approx \perp)$	$(u \vee v, \perp, v \approx \perp)$
$(u \rightarrow v, \top, u \approx \perp \vee v \approx \top)$	$(u \rightarrow v, \perp, u \approx \top)$	$(u \rightarrow v, \perp, v \approx \perp)$
$(u \approx^\tau v, \top, u \approx v)$	$(u \approx^\tau v, \perp, u \not\approx v)$	
$(u \not\approx^\tau v, \top, u \not\approx v)$	$(u \not\approx^\tau v, \perp, u \approx v)$	
$(\neg u, \top, u \approx \perp)$	$(\neg u, \perp, u \approx \top)$	

$$\frac{C\langle u \rangle}{C\langle \perp \rangle \vee u \approx \top} \text{IGBOOLHOIST} \quad \frac{C\langle u \rangle}{C\langle \top \rangle \vee u \approx \perp} \text{IGLOOBHOIST}$$

each with the following side conditions:

1. u is of Boolean type;
2. u is neither \top nor \perp ;
3. the position of u is eligible in C ;
4. the occurrence of u is not in a literal of the form $u \approx \perp$ or $u \approx \top$.

$$\frac{\overbrace{C' \vee \perp \approx \top}^C}{C'} \text{IGFALSEELIM}$$

with the following side conditions:

1. $\perp \approx \top$ is strictly eligible in C .

$$\frac{\overbrace{C' \vee s \approx s'}^C}{C' \vee s \text{ diff}_{u,v}^{\tau,v} \approx s' \text{ diff}_{u,v}^{\tau,v}} \text{IGARGCONG}$$

with the following side conditions:

1. s is of type $\tau \rightarrow v$;
2. u, v are ground terms of type $\tau \rightarrow v$;
3. $s \approx s'$ is strictly eligible in C .

$$\frac{C\langle u \rangle}{C\langle v \rangle \vee u \text{ diff}_{u,v}^{\tau,v} \not\approx v \text{ diff}_{u,v}^{\tau,v}} \text{IGEXT}$$

with the following side conditions:

1. the position of u is eligible in $C\langle u \rangle$;
2. the type of u is $\tau \rightarrow v$;
3. v is a ground term of type $\tau \rightarrow v$;
4. $u \succ v$;

$$\frac{}{u \text{ diff}_{u,v}^{\tau,v} \not\approx u \text{ diff}_{u,v}^{\tau,v} \vee u s \approx v s} \text{IGDIFF}$$

with the following side conditions:

1. τ and v are ground types;
2. u, v are ground terms of type $\tau \rightarrow v$;
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 $\mathcal{T}_{\text{ground}}(\Sigma_H)$ and on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_H)$. Such a relation \succ is an *admissible term order for GInf* if it fulfills the following properties:

- (O1)_G the relation \succ on ground terms is a well-founded total order;
- (O2)_G ground compatibility with yellow contexts: $s' \succ s$ implies $t \ll s' \gg \succ t \ll s \gg$ for ground terms s, s' , and t ;
- (O3)_G ground yellow subterm property: $t \ll s \gg \succeq s$ for ground terms s and t ;
- (O4)_G $u \succ \perp \succ \top$ for all ground terms $u \notin \{\top, \perp\}$;
- (O5)_G $u \succ u \text{ diff}(\tau, v)(s, t)$ for all ground terms $s, t, u : \tau \rightarrow v$.
- (O6)_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 C_G instead of C_{IG} .
- The rules are prefixed by G instead of IG .
- GARGCONG uses $\text{diff}(\tau, v)(u, v)$ instead of $\text{diff}_{u,v}^{\tau,v}$.
- GEXT uses $\text{diff}(\tau, v)(u, v)$ instead of $\text{diff}_{u,v}^{\tau,v}$.
- GDIFF uses $\text{diff}(\tau, v)(u, v)$ instead of $\text{diff}_{u,v}^{\tau,v}$.

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 C_F .

Definition 4.11 (Inference Redundancy). Given $\iota \in FInf$ and $N \subseteq C_F$, let $\iota \in FRed_1(N)$ if ι is a DIFF inference and $N \models_{o\lambda} \text{concl}(\iota)$ or if ι is not a DIFF inference and $\{E \in N \mid E \prec \text{mprem}(\iota)\} \models_{o\lambda} \text{concl}(\iota)$.

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 C_{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_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 $\mathcal{T}_{\text{ground}}(\Sigma_I)$ and \mathcal{T}_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. \square

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

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

Lemma 4.14. The mapping \mathcal{F} is a bijection between $IGInf^{\succ, igsel}$ and $FInf^{\succ_F, \mathcal{F}(igsel)}$, where $\mathcal{F}(igsel)$ 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 $\mathcal{F}(igsel)$ (Definition 2.23).
- The concepts of eligibility correspond; i.e., a literal L of a clause $C \in \mathcal{C}_{IG}$ 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 $C \in \mathcal{C}_{IG}$ 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. \square

Definition 4.15 (Inference Redundancy). Given $\iota \in IGInf^{\succ,igsel}$ and $N \subseteq \mathcal{C}_{IG}$, let $\iota \in IGRed_1(N)$ if $\mathcal{F}(\iota) \in FRed_1(\mathcal{F}(N))$ (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. $IGInf^{\succ,igsel}$. Then $\mathcal{F}(N)$ is saturated up to redundancy w.r.t. $FInf^{\succ_{\mathcal{F}},\mathcal{F}(igsel)}$.*

Proof. By Lemma 4.14 and Definition 4.15. \square

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}_G .

Since mapping \mathcal{J} 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 $\mathcal{T}_{\text{ground}}(\Sigma_H)$ and on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_H)$. We define a relation $\succ_{\mathcal{J}}$ on $\mathcal{T}_{\text{ground}}(\Sigma_I)$ and on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_I)$ as $d \succ_{\mathcal{J}} e$ if and only if $\mathcal{J}^{-1}(d) \succ \mathcal{J}^{-1}(e)$ 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 $\succ_{\mathcal{J}}$ is an admissible term order for $IGInf$ (Definition 4.8).*

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

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

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

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

$$\frac{\mathcal{J}(\text{prems}(\iota))}{\mathcal{J}(\text{concl}(\iota))}$$

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

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

Definition 4.22 (Inference Redundancy). Let $N \subseteq \mathcal{C}_G$ and $\iota \in GInf$. We define $\iota \in GRed_1$ if $\mathcal{I}(\iota) \in IGRed_1(\mathcal{I}(N))$.

Lemma 4.23. Let $N \subseteq \mathcal{C}_G$ be saturated up to redundancy w.r.t. $GInf^{\succ, g_{sel}}$. Then $\mathcal{I}(N)$ is saturated up to redundancy w.r.t. $IGInf^{\succ_{\mathcal{I}}, \mathcal{I}(g_{sel})}$.

Proof. By Lemma 4.21 and Definition 4.22. \square

4.3.4. Full Higher-Order Level. In this subsubsection, let \succ be an admissible term order (Definition 2.12) and let h_{sel} 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 (O1)_G to (O6)_G. \square

The selection function is transferred as follows:

Definition 4.25. Let $N \subseteq \mathcal{C}_H$. 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 $\mathcal{G}(h_{sel}, N)$ as a selection function that selects the literals of $C \in \mathcal{G}(N)$ corresponding to the h_{sel} -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 $t, s \in \mathcal{T}(\Sigma_H)$, 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). \square

Lemma 4.27 (Lifting of Maximality Conditions). Let $C \in \mathcal{C}_H$. 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, $K\theta \not\prec L\theta = L_0$. 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 $C\theta$. \square

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

- Let L_G be a literal in C_G that is (strictly) eligible w.r.t. $\mathcal{G}(\text{hsel}, N)$. Then there exists a literal L_H in C_H such that $L_G = L_H\theta$ 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 $L_G.s_G.p_G$ be a green position of C_G that is eligible w.r.t. $\mathcal{G}(\text{hsel}, N)$. Then there exists a green position $L_H.s_H.p_H$ of C_H such that
 - $L_G = L_H\theta$;
 - $s_G = s_H\theta$;
 - * $p_G = p_H$, or
 - * $p_G = p_H.q$ for some nonempty q , the subterm u_H at position $L_H.s_H.p_H$ of C_H is variable-headed, and $u_H\theta$ is nonfunctional; and
 - given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in C_H , $L_H.s_H.p_H$ 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. $\mathcal{G}(\text{hsel}, N)$. By the definition of eligibility (Definition 2.15), there are two ways to be (strictly) eligible:

- L_G is selected by $\mathcal{G}(\text{hsel}, N)$. By Definition 4.25, there exists a literal L_H selected by hsel such that $L_G = L_H\theta$. 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 $C_G = C_H\theta = C_H\sigma\zeta$, by Lemma 4.27, there exists a literal $L_H \in C_H$ such that $L_H\sigma$ is (strictly) maximal in $C_H\sigma$. By Definition 2.15, L_H is (strictly) eligible in $C_H[\![S]\!]$ w.r.t. σ .

For the second part of the lemma, let $L_G.s_G.p_G$ be a green position of C_G that is eligible w.r.t. $\mathcal{G}(\text{hsel}, N)$. By Definition 2.15, the literal L_G is of the form $s_G \approx t_G$ with $s_G \succ t_G$ 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 $L_G = L_H\theta$. Then L_H must be of the form $s_H \approx t_H$ with $s_H = s_H\theta$ and $t_H = t_H\theta$. Since $s_G \succ t_G$, we have $s_H \not\leq t_H$. By Definition 2.15, every green position in $L_H.s_H$ is eligible in C_H w.r.t. σ and hsel .

It remains to show that there exists a green position $L_H.s_H.p_H$ in C_H such that either $p_G = p_H$ or $p_G = p_H.q$ for some nonempty q , the subterm u_H at position $L_H.s_H.p_H$ of C_H is variable-headed, and $u_H\theta$ is nonfunctional.

Since p_G is a green position of $s_G = s_H\theta$, 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 $p_H = p_G$. In the second case, let u_H be that variable-headed term and let p_H be its position. Then $p_H.q = p_G$ 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 $u_H\theta$, cannot be functional. \square

Lemma 4.29 (Lifting Lemma). *Let $N \subseteq C_H$ be saturated up to redundancy w.r.t. $H\text{Inf}^{\succ, \text{hsel}}$. Then $\mathcal{G}(N)$ is saturated up to redundancy w.r.t. $G\text{Inf}^{\succ, \mathcal{G}(\text{hsel}, N)}$.*

Proof. Let ι_G be a $GInf$ inference from $\mathcal{G}(N)$. We must show that $\iota_G \in GRed_1(\mathcal{G}(N))$. By Definitions 4.22, 4.15, 4.11, and Lemma 4.5, it suffices to show that

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

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 $\theta_1, \dots, \theta_{m+1}$ such that applying $\theta_1, \dots, \theta_m$ to $\text{prems}(\iota_H)$ yields $\text{prems}(\iota_G)$ and applying θ_{m+1} to $\text{concl}(\iota_H)$ yields $\text{concl}(\iota_G)$. (**)

By Lemmas 4.21 and 4.14, $\mathcal{F}(\mathcal{I}(\iota_G))$ is a valid $FInf^{\succ_{\mathcal{F}}, \mathcal{F}(\mathcal{I}(\mathcal{G}(hsel, N)))}$ inference. By Lemma 4.5, $\succ_{\mathcal{F}} = \succ_{\mathcal{F}}$ and $\mathcal{F}(\mathcal{I}(\mathcal{G}(hsel, N))) = \mathcal{F}(\mathcal{G}(hsel, N))$. Moreover, Lemma 4.5 tells us that $\mathcal{F}(\mathcal{I}(\iota_G))$ can also be obtained by applying \mathcal{F} directly to premisses and conclusion of ι_G . Therefore, ι_H is rooted in $FInf$ for $(\theta_1, \dots, \theta_{m+1})$ (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

$$\frac{\overbrace{D'_G \vee t_G \approx t'_G}^{D_G} \quad C_G \langle t_G \rangle}{D'_G \vee C_G \langle t'_G \rangle} \text{IGSUP}$$

Let $C_H = \mathcal{G}_N^{-1}(C_G)$, and let θ be a grounding substitution such that $C_G = C_H\theta$. By condition 4 of GSUP, the position $L_G.s_G.p_G$ of t_G is eligible in C_G . By Lemma 4.28, there exists a green position $L_H.s_H.p_H$ of C_H such that

- $L_G = L_H\theta$;
- $s_G = s_H\theta$;
- given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in C_H , $L_H.s_H.p_H$ 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: $p_G = p_H$.

CASE 1A: The subterm at position $L_H.s_H.p_H$ 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 $C_H\theta[x \mapsto x]$ either. By condition 1 of GSUP, t_G is nonfunctional and so x is nonfunctional. Thus, all occurrences of x in $C_H\theta[x \mapsto x]$ are in green positions. Thus, $C_H\theta[x \mapsto t'_G]$ results from $C_H\theta = C_G$ by replacing t_G with t'_G at green positions. Thus, by the T_G -analogue of Lemma 4.3, $\mathcal{F}(C_G)$ results from $\mathcal{F}(C_H\theta[x \mapsto t'_G])$ by replacing $\mathcal{F}(t_G)$ with $\mathcal{F}(t'_G)$. Therefore, $\{\mathcal{F}(t_G) \approx \mathcal{F}(t'_G), \mathcal{F}(C_H\theta[x \mapsto t'_G])\} \models_{\text{o}\lambda} \mathcal{F}(C_G)$ and thus

$$\{\mathcal{F}(D_G), \mathcal{F}(C_H\theta[x \mapsto t'_G])\} \models_{\text{o}\lambda} \mathcal{F}(D'_G \vee C_G \langle t'_G \rangle)$$

By condition 3 of GSUP, $D_G \prec \text{mprem}(\iota_G)$, and thus $\mathcal{F}(D_G) \prec_{\mathcal{F}} \mathcal{F}(\text{mprem}(\iota_G))$. By Condition 2 of GSUP, $t_G \succ t'_G$, and thus by (O2) $_{\mathcal{F}}$, $\mathcal{F}(\text{mprem}(\iota_G)) = \mathcal{F}(C_H\theta[x \mapsto t_G]) \succ_{\mathcal{F}} \mathcal{F}(C_H\theta[x \mapsto t'_G])$. Therefore,

$$\{E \in \mathcal{F}(\mathcal{G}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(\text{mprem}(\iota_G))\} \models_{\text{o}\lambda} \mathcal{F}(\text{concl}(\iota_G))$$

and thus $(*)$.

CASE 1B: The subterm u_H at position $L_H.s_H.p_H$ 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 $D_H = \mathcal{G}_N^{-1}(D_G)$, and $C_H = \mathcal{G}_N^{-1}(C_G)$. Let ρ be the grounding substitution such that $D_G = D_H\rho$ and let θ be the grounding substitution such that $C_G = C_H\theta$.

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

Condition 5 of GSUP states that $t_G \approx t'_G$ is strictly eligible in D_G . Let $D_H = D'_H \vee t_H \approx t'_H$, where $t_H \approx t'_H$ is the literal that Lemma 4.28 guarantees to be strictly eligible w.r.t. any suitable σ (condition 6 of SUP), with $t_H\rho = t_G$ and $t'_H\rho = t'_G$. Condition 6 of GSUP states that if t'_G is Boolean, then $t'_G = \top$. 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 $L_G = L_H\theta$, $s_G = s_H\theta$, and $p_G = p_H$, we have $u_H\theta = t_G = t_H\rho$. 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 $t_H \equiv u_H$. By Definition 2.11, there exists a unifier $\sigma \in \text{CSU}(t_H \equiv u_H)$ 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 $u_H\sigma\zeta = u_H\theta = t_G$, $u_H\sigma$ is nonfunctional (condition 3 of SUP).

By condition 2 of GSUP, $t_H\sigma\zeta = t_G \succ t'_G = t'_H\sigma\zeta$, and thus by Lemma 4.26, $t_H\sigma \not\leq t'_H\sigma$ (condition 4 of SUP).

Moreover, $\text{concl}(\iota_H)\zeta = \text{concl}(\iota_G)$. Thus, we can apply (**).

CASE 2: $p_G = p_H.q$ for some nonempty q , the subterm u_H at position $L_H.s_H.p_H$ of C_H is variable-headed, and $u_H\theta$ is nonfunctional.

CASE 2A: The subterm at position $L_H.s_H.p_H$ 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 $L_H.s_H.p_H$ 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 $D_H = \mathcal{G}_N^{-1}(D_G)$, and $C_H = \mathcal{G}_N^{-1}(C_G)$. Let ρ be the grounding substitution such that $D_G = D_H\rho$ and let θ be the grounding substitution such that $C_G = C_H\theta$.

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 $v = \lambda(u_H\theta)\langle n \rangle_q$, 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, $(z t_H)\theta' = v(t_H\rho) = v t_G = (u_H\theta)\langle t_G \rangle_q = u_H\theta = u_H\theta'$. So θ' is a unifier of $z t_H$ and u_H . Thus, by definition of CSU (Definition 2.11), there exists a unifier $\sigma \in \text{CSU}(z t_H \equiv u_H)$ and a substitution ζ such that $x\sigma\zeta = x\theta'_H$ for all relevant variables x (condition 1 of FLUIDSUP).

The assumption of Case 2 tells us that $u_H\theta = u_H\sigma\zeta$ is nonfunctional. It follows that $u_H\sigma$ is nonfunctional (condition 3 of FLUIDSUP).

By condition 2 of GSUP, $t_H\sigma\zeta = t_G \succ t'_G = t'_H\sigma\zeta$, and thus by Lemma 4.26, $t_H\sigma \not\leq t'_H\sigma$ (condition 4 of FLUIDSUP).

By condition 2 of GSUP, $t_H\rho = t_G \neq t'_G = t'_H\rho$. Thus, $(z t_H)\sigma\zeta = v(t_H\rho) = u_H\theta\langle t_H\rho \rangle_q \neq u_H\theta\langle t'_H\rho \rangle_q = v(t'_H\rho) = (z t'_H)\sigma\zeta$. So, $(z t'_H)\sigma \neq (z t_H)\sigma$ (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, $\text{concl}(\iota_H)\zeta = (D'_H \vee C_H \langle z t'_H \rangle_{p_H})\sigma\zeta = D'_G \vee C_G \langle (u_H\theta) \langle t'_G \rangle_q \rangle_{p_H} = D'_G \vee C_G \langle t'_G \rangle_{p_G} = \text{concl}(\iota_G)$. Thus, we can apply (**).

GEQRES: Assume that ι_G is a GEQRES inference

$$\frac{\overbrace{C'_G \vee u_G \not\approx u_G}^{C_G}}{C'_G} \text{GEQRES}$$

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

Moreover, $\text{concl}(\iota_H)\zeta = \text{concl}(\iota_G)$. Thus, we can apply (**).

GEQFACT: Assume that ι_G is a GEQFACT inference

$$\frac{\overbrace{C'_G \vee u_G \approx v'_G \vee u_G \approx v_G}^{C_G}}{C'_G \vee v_G \not\approx v'_G \vee u_G \approx v_G} \text{GEQFACT}$$

Let $C_H = \mathcal{G}_N^{-1}(C_G)$, and let θ be a grounding substitution such that $C_G = C_H\theta$. Condition 1 of GEQFACT states that $u_G \approx v_G$ is maximal in C_G . Let $u_H \approx v_H$ 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 $u_H\theta = u_G$ and $v_H\theta = v_G$. Choose C'_H , u'_H , and v'_H such that $C_H = C'_H \vee u'_H \approx v'_H \vee u_H \approx v_H$, $C'_H\theta = C'_G$, $u'_H\theta = u_G$, and $v'_H\theta = v_G$.

Then θ is a unifier of u_H and u'_H , and thus there exists a unifier $\sigma \in \text{CSU}(u_H \equiv u'_H)$ 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, $u_H\sigma\zeta = u_G \succ v_G = v_H\sigma\zeta$. By Lemma 4.26, $u_H\sigma \not\leq v_H\sigma$ (condition 4 of EQFACT).

Moreover, $\text{concl}(\iota_H)\zeta = \text{concl}(\iota_G)$. Thus, we can apply (**).

GCLAUSIFY: Assume ι_G is a GCLAUSIFY inference

$$\frac{\overbrace{C'_G \vee s_G \approx t_G}^{C_G}}{C'_G \vee D_G} \text{GCLAUSIFY}$$

with τ_G being the type and u_G and v_G being the terms used for condition 2. Let $C_H = \mathcal{G}_N^{-1}(C_G)$, and let θ be a grounding substitution such that $C_G = C_H\theta$.

Condition 1 of GCLAUSIFY is that $s_G \approx t_G$ is strictly eligible in C_G . Let $C_H = C'_H \vee s_H \approx t_H$, where $s_H \approx t_H$ is the literal that Lemma 4.28 guarantees to be strictly eligible w.r.t. any suitable σ (condition 2 of CLAUSIFY), with $s_H\theta = s_G$ and $t_H\theta = t_G$.

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 $\theta_{\mathbf{T}}$ and θ_{\perp} that coincide with θ , except for mapping s_H to \mathbf{T} and \perp , respectively. Since the semantics of $\models_{\circ\lambda}$ interpret Booleans, we have $\{\mathcal{F}(C_H\theta_{\mathbf{T}}), \mathcal{F}(C_H\theta_{\perp})\} \models_{\circ\lambda} \mathcal{F}(C_H\theta)$ because s_H does not appear in parameters in C_H . By (O4)_F and (O2)_F, $\mathcal{F}(C_H\theta_{\mathbf{T}}) \prec \mathcal{F}(C_H\theta_{\perp}) \prec \mathcal{F}(C_H\theta)$. 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'_H, t'_H, D_H) listed for CLAUSIFY such that $(s'_H\rho, t'_H\rho, D_H\rho) = (s_G, t_G, D_G)$ with $\rho = \{\alpha \mapsto \tau_G, x \mapsto u_G, y \mapsto v_G\}$ is the triple used for ι_G (condition 4 of CLAUSIFY).

Moreover, we observe that $s_H\theta = s_G = s'_H\rho$ and $t_H\theta = t_G = t'_H\rho$. 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 $s_H \equiv s'_H$ and $t_H \equiv t'_H$. So there exists a unifier $\sigma \in \text{CSU}(s_H \equiv s'_H, t_H \equiv t'_H)$ (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 $\text{concl}(\iota_H)\zeta = \text{concl}(\iota_G)$. Thus, we can apply (**).

GBOOLHOIST: Analogous to GSUP, using the substitutions $\theta_{\mathbf{T}}$ and θ_{\perp} 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

$$\frac{C_G \langle u_G \rangle}{C_G \langle v_G \rangle \vee u_G \text{ diff} \langle \tau_G, v_G \rangle (u_G, v_G) \not\approx v_G \text{ diff} \langle \tau_G, v_G \rangle (u_G, v_G)} \text{IGExt}$$

Let $C_H = \mathcal{G}_N^{-1}(C_G)$, and let θ be a grounding substitution such that $C_G = C_H\theta$. By condition 1 of GEXT, the position $L_G.s_G.p_G$ of u_G is eligible in C_G . By Lemma 4.28, there exists a green position $L_H.s_H.p_H$ of C_H such that

- $L_G = L_H\theta$;
- $s_G = s_H\theta$;
- given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in C_H , $L_H.s_H.p_H$ 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: $p_G = p_H$. Then we construct an EXT inference ι_H from C_H .

Let u_H be the subterm of C_H at position $L_H.s_H.p_H$. Since $p_G = p_H$, we have $u_H\theta = u_G$. By condition 2 of GEXT, u_G is functional and thus there exists a most general type substitution σ such that $u_H\sigma$ 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 $u_H\sigma$ (condition 2 of EXT). Let $\zeta' = \zeta[y \mapsto v_G]$. Then, $\text{concl}(\iota_H)\zeta' = \text{concl}(\iota_G)$. Thus, we can apply (**).

CASE 2: $p_G = p_H.q$ for some nonempty q , the subterm u_H at position $L_H.s_H.p_H$ of C_H is variable-headed, and $u_H\theta$ is nonfunctional.

Then we construct a FLUIDSUP inference ι_H from $C_H = \mathcal{G}_N^{-1}(C_G)$.

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 $\alpha \rightarrow \beta$, and let z be a fresh variable of function type from $\alpha \rightarrow \beta$ to the type of u_H (condition 3 of FLUIDEXT).

Let $v = \lambda(u_H\theta)\langle n \rangle_q$, where n is the appropriate De Bruijn index to refer to the initial λ . We define a substitution θ' that coincides with θ on all variables except $\alpha\theta' = \tau_G$, $\beta\theta' = v_G$, $z\theta' = v$, $x\theta' = u_G$, and $y\theta' = v_G$. Then, $(z x)\theta' = v u_G = (u_H\theta)\langle u_G \rangle_q = u_H\theta = u_H\theta'$. So θ' is a unifier of $z x$ and u_H . Thus, by definition of CSU (Definition 2.11), there exists a unifier $\sigma \in \text{CSU}(z x \equiv u_H)$ 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 $u_H\theta = u_H\sigma\zeta$ is nonfunctional. It follows that $u_H\sigma$ is nonfunctional (condition 2 of FLUIDEXT).

By condition 4 of GEXT, $u_G \neq v_G$. Thus, $(zx)\sigma\zeta = vu_G = (u_H\theta)\langle u_G \rangle_q \neq (u_H\theta)\langle v_G \rangle_q = v v_G = (zy)\sigma\zeta$. So, $(zx)\sigma \neq (zy)\sigma$ (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,

$$\begin{aligned} \text{concl}(\iota_H)\zeta &= (C_H\langle z y \rangle \vee x \text{ diff}\langle \alpha, \beta \rangle(x, y) \not\approx y \text{ diff}\langle \alpha, \beta \rangle(x, y))\sigma\zeta \\ &= C_G\langle (u_H\theta)\langle v_G \rangle_q \rangle_{p_H} \vee u_G \text{ diff}\langle \tau_G, v_G \rangle(u_G, v_G) \not\approx v_G \text{ diff}\langle \tau_G, v_G \rangle(u_G, v_G) \\ &= C_G\langle v_G \rangle_{p_G} \vee u_G \text{ diff}\langle \tau_G, v_G \rangle(u_G, v_G) \not\approx v_G \text{ diff}\langle \tau_G, v_G \rangle(u_G, v_G) \\ &= \text{concl}(\iota_G) \end{aligned}$$

Thus, we can apply (**).

GDIFF: Assume that ι_G is a GDIFF inference

$$\frac{}{u \text{ diff}\langle \tau, v \rangle(u, v) \not\approx u \text{ diff}\langle \tau, v \rangle(u, v) \vee u s \approx v s} \text{GDIFF}$$

Then we use the following DIFF inference ι_H :

$$\frac{}{y (\text{diff}\langle \alpha, \beta \rangle(y, z)) \not\approx z (\text{diff}\langle \alpha, \beta \rangle(y, z)) \vee y x \approx z x} \text{DIFF}$$

Clearly, $\text{concl}(\iota_H)\theta = \text{concl}(\iota_G)$ for $\theta = \{\alpha \mapsto \tau, \beta \mapsto v, y \mapsto u, z \mapsto v, x \mapsto s\}$. Thus, we can apply (**). \square

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 C_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 $R \models_{fol} s \approx t$ 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 $\perp \notin N$. By well-founded induction, we define term rewrite systems R_e and Δ_e for all ground clauses and ground terms $e \in T_F \cup C_F$ and finally a term rewrite system R_N . As our well-founded order on $T_F \cup C_F$, 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 $\{\{s\}\}$.

- ($\Delta 1$) LOGICAL BOOLEAN REWRITES: Given a term s , let $\Delta_s = \{s \rightarrow t\}$ if
– (s, t) is one of the following:

$$\begin{array}{lllll} (\neg \perp, \top) & (\top \wedge \top, \top) & (\top \vee \top, \top) & (\top \rightarrow \top, \top) & (u \approx^\tau u, \top) \\ (\neg \top, \perp) & (\top \wedge \perp, \perp) & (\top \vee \perp, \perp) & (\top \rightarrow \perp, \perp) & (u \approx^\tau v, \perp) \text{ with } u \neq v \\ (\perp \wedge \top, \perp) & (\perp \vee \top, \top) & (\perp \rightarrow \top, \top) & & (u \not\approx^\tau u, \perp) \\ (\perp \wedge \perp, \perp) & (\perp \vee \perp, \perp) & (\perp \rightarrow \perp, \top) & & (u \not\approx^\tau v, \top) \text{ with } u \neq v \end{array}$$

- s is irreducible w.r.t. R_s .
- ($\Delta 2$) BACKSTOP BOOLEAN REWRITES: Given a clause C , let $\Delta_C = \{s \rightarrow \perp\}$ if
– $C = s \approx \perp$;
– $s \notin \{\perp, \top\}$;
– s is irreducible w.r.t. R_C .
- ($\Delta 3$) FUNCTION REWRITES: Given a clause C , let $\Delta_C = \{\mathcal{F}(u) \rightarrow \mathcal{F}(v)\}$ if
– $C = \mathcal{F}(u) \approx \mathcal{F}(v)$ for functional terms u and v ;
– $\mathcal{F}(u) \succ \mathcal{F}(v)$;
– $\mathcal{F}(u \text{ diff}_{s,t}^{\tau,v}) \leftrightarrow_{R_C}^* \mathcal{F}(v \text{ diff}_{s,t}^{\tau,v})$ for all s, t ;
– $\mathcal{F}(u)$ is irreducible w.r.t. R_C .
- ($\Delta 4$) PRODUCED REWRITES: Given a clause C , let $\Delta_C = \{s \rightarrow t\}$ 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) $R_C \not\models_{fol} C$;
(CC10) $R_C \cup \{s \rightarrow t\} \not\models_{fol} C'$.

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

- ($\Delta 5$) For all other terms and clauses e , Let $\Delta_e = \emptyset$.

Let $R_e = \bigcup_{f \prec e} \Delta_f$. Let $R_N = \bigcup_{e \in T_F \cup C_F} \Delta_e$.

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 (O4)_F.

To show the absence of critical pairs, suppose that there exists a critical pair $s \rightarrow t$ and $s' \rightarrow t'$ in R_N , originating from Δ_e and $\Delta_{e'}$ respectively, for some $e, e' \in \mathcal{T}_F \cup \mathcal{C}_F$. Without loss, we assume $e \succ e'$. Inspecting the rules of Definition 4.30, it follows that $s \succeq s'$. By the subterm property (O3)_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. \square

Lemma 4.32. *The normal form of any ground Boolean term w.r.t. R_N is \top or \perp .*

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

It remains to show that any ground Boolean term s reduces to \top or \perp . We prove the claim by induction on s w.r.t. \succ . If $s = \top$ or $s = \perp$, we are done. Otherwise, consider the rule $(\Delta 2)$ for $C = s \approx \perp$. 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 \top or \perp . Therefore, s reduces to \top or \perp . \square

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

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

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, $R_N \models_{\text{fol}} L$.

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 $s \approx s'$ be the maximal literal in C with $s \succeq s'$. We have $s \succ t$ if $s \approx s'$ is positive and $s \succeq t$ if $s \approx s'$ 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 $R_N \models_{\text{fol}} L$. \square

Lemma 4.34. *If a clause $C = C' \vee s \approx t \in \mathcal{C}_F$ produces $s \rightarrow t$, then $R_N \not\models_{\text{fol}} C'$.*

Proof. By (CC5) and (CC6), all terms in C are smaller or equal to s . By (CC10), we have $R_C \cup \{s \rightarrow t\} \not\models C'$. The other rules $R_N \setminus (R_C \cup \{s \rightarrow t\})$ 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, $R_C \cup \{s \rightarrow t\} \not\models_{\text{fol}} C'$ implies $R_N \not\models_{\text{fol}} C'$. \square

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

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

By definition of \mathcal{F} , since u is functional, $\mathcal{F}(u) = \text{fun}_u$. 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 $(\Delta 3)$. One of the conditions of $(\Delta 3)$ then yields the claim. \square

Lemma 4.36. *Let u and v be higher-order ground terms of type $\tau \rightarrow v$. If $\mathcal{F}(u \text{ diff}_{s,t}^{\tau,v}) \leftrightarrow_{R_N}^* \mathcal{F}(v \text{ diff}_{s,t}^{\tau,v})$ for all s, t , then $\mathcal{F}(u) \leftrightarrow_{R_N}^* \mathcal{F}(v)$.*

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

We want to show that $\mathcal{F}(u) \leftrightarrow_{R_N}^* \mathcal{F}(v)$ —i.e., that $\mathcal{F}(u') = \mathcal{F}(v')$. To derive a contradiction, we assume that $\mathcal{F}(u') \neq \mathcal{F}(v')$. Without loss of generality, we may assume that $\mathcal{F}(u') \succ \mathcal{F}(v')$. Then, using $(O5)_F$, all conditions of $(\Delta 3)$ are satisfied for the rule $\mathcal{F}(u') \rightarrow \mathcal{F}(v')$, contradicting the fact that $\mathcal{F}(u')$ is a normal form. \square

Lemma 4.37. *R_N is a $\models_{o\lambda}$ -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 \top and \perp . The rule $(\Delta 1)$ ensures that the symbols $\neg, \wedge, \vee, \rightarrow, \approx^\tau, \not\models^\tau$ 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 $\mathcal{F}(\mathcal{I}(u) \text{ diff}_{s,t}^{\tau,v}) = \mathcal{F}(u \text{ diff}(\tau, v)(s, t))$ for all $u, s, t \in \mathcal{T}_{\text{ground}}(\Sigma_H)$. Since \mathcal{I} 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. \square

We employ a variant of Bachmair and Ganzinger’s framework of reducing counter-examples [2, Sect. 4.2]. Let $N \subseteq C_F$ with $\perp \notin N$. A clause $C_0 \in C_F$ is called a *counterexample* if $R_N \not\models_{\text{fol}} C_0$. 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 $N \subseteq C_F$ 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 $s \rightarrow s' \in R_C$. 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 $s \rightarrow s'$. We make a case distinction on which case of Definition 4.30 the rule $s \rightarrow s'$ originates from:

- $(\Delta 1)$ Then the root of s is a logical symbol and $s \notin \{\top, \perp\}$. By Lemma 4.32, R_N reduces s to \top or to \perp .
- First consider the case where the position p in C is in a literal of the form $s \approx \top$ or $s \approx \perp$. Then FCLAUSIFY 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 \perp) or PFLOOBHOIST (if s reduces to \top). 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 \perp and $s \notin \{\top, \perp\}$. Due to the presence of the rule $s \rightarrow \perp$ in R_C , C must be larger than $s \approx \perp$. 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 $s \approx \perp$ either because $s \approx \perp$ 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 $v = \mathcal{F}^{-1}(s \downarrow_{R_N})$. Then $\mathcal{F}(u) \leftrightarrow_{R_N}^* \mathcal{F}(v)$. By Lemma 4.35, $\mathcal{F}(u \text{ diff}_{u,v}^{\tau,v}) \leftrightarrow_{R_N}^* \mathcal{F}(v \text{ diff}_{u,v}^{\tau,v})$. Since R_N is oriented by \succ , we have $\mathcal{F}(u) = s \succ s \downarrow_{R_N} = \mathcal{F}(v)$. 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 (O5)_F and because $\mathcal{F}(u) \succ \mathcal{F}(v)$.
- (Δ4) Then some clause $D \vee s \approx s'$ smaller than C produces the rule $s \rightarrow s'$. 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 $R_{D \vee s \approx s'} \cup \{s \rightarrow s'\} \models_{\text{fol}} D$, in contradiction to (CC10).
- If s' is Boolean, then $s' = \top$ by (CC4).

As $D \vee s \approx s'$ is productive, $R_N \not\models_{\text{fol}} D$ 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 (O2)_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 . \square

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 $R_N \not\models_{\text{fol}} C$, we have $R_C \not\models_{\text{fol}} C$ by Lemma 4.33. Therefore, $R_C \not\models_{\text{fol}} s \not\approx s'$ and $R_C \models_{\text{fol}} s \approx s'$. 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 $R_N \not\models_{\text{fol}} C$, 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 $s = \perp$. Then, since $s \succ s'$, we have $s' = \top$ by $(O4)_F$. 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 $(O5)_F$. By Lemma 4.36, there must be at least one choice of u and v in the FARGCONG rule such that the conclusion is a counterexample.
- Case 2.5: L is strictly eligible and $s \neq \perp$ is nonfunctional and not reducible by R_C . Since $R_N \not\models_{\text{fol}} C$, C cannot be productive. So at least one of the conditions of $(\Delta 4)$ 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 $(\Delta 1)$ applies. The condition in $(\Delta 1)$ that any Boolean arguments of s must be \top or \perp 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 $(\Delta 1)$ 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 $s \neq \top, \perp$ by $(O4)_F$. Moreover, $s' \succeq \perp$, and thus $C \succeq s \approx \perp$. Since s is not reducible by R_C , it is also irreducible by $R_{s \approx \perp} \subseteq R_C$. So $(\Delta 2)$ triggers and sets $\Delta_{s \approx \perp} = \{s \rightarrow \perp\}$. Since s is not reducible by R_C , we must have $C = s \approx \perp$. But then C is true 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 $R_C \cup \{s \rightarrow s'\} \models_{\text{fol}} C'$, where C' is the subclause of C with L removed. However, $R_C \not\models_{\text{fol}} C$, and therefore, $R_C \not\models_{\text{fol}} C'$. Thus, we must have $C' = C'' \vee r \approx t$ for some terms r and t , where $R_C \cup \{s \rightarrow s'\} \models_{\text{fol}} r \approx t$ and $R_C \not\models_{\text{fol}} r \approx t$. So $r \neq t$ and without loss of generality we assume $r \succ t$. Moreover $s \rightarrow s'$ must participate in the normalization of r or t by $R_C \cup \{s \rightarrow s'\}$. Since $s \approx s'$ is maximal in C by (CC6), $r \preceq s$. So the rule $s \rightarrow s'$ can be used only as the first step in the normalization of r . Hence $r = s$ and $R_C \models_{\text{fol}} s' \approx t$. Then FEQFACT reduces the counterexample. \square

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 $FRed_I$, and $\perp \notin N$. Then $R_N \models_{o\lambda} N$.*

Proof. By Lemma 4.37, it suffices to show that $R_N \models_{\text{fol}} N$. For a proof by contradiction, we assume that $R_N \not\models_{\text{fol}} N$. Then N contains a minimal counterexample, i.e., a clause C with $R_N \not\models_{\text{fol}} C$. 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 $concl(\iota)$ that is smaller than C and false in R_N . By saturation up to redundancy, $\iota \in FRed_I$. By Definition 4.11, we have $\{E \in N \mid E \prec C\} \models_{o\lambda} concl(\iota)$. 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 $concl(\iota)$ is true in R_N , a contradiction. \square

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 C_{IG} , and let $N \subseteq C_{IG}$ such that N is saturated up to redundancy w.r.t. $IGInf$ and $\perp \notin N$. We write R for the term rewrite system $R_{\mathcal{F}(N)}$ constructed in the previous subsubsection w.r.t. $\succ_{\mathcal{F}}$ and $\mathcal{F}(igsel)$. We write $t \sim s$ for $\mathcal{F}(t) \leftrightarrow_R^* \mathcal{F}(s)$, where $t, s \in T_{\text{ground}}(\Sigma_I)$.

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 $s, s' \in T_{\text{ground}}(\Sigma_I)$. Let $u \in T_{\text{ground}}(\Sigma_I)$. Then $s u \sim s' u$.*

Proof. Consider the following inference ι :

$$\frac{s \text{ diff}_{s,s'}^{\tau,v} \not\approx s' \text{ diff}_{s,s'}^{\tau,v} \vee s u \approx s' u}{\text{IGDIFF}}$$

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

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

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 $s \in T(\Sigma_I)$, 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 $s \in T^\lambda(\Sigma_I)$. Here, for $t_1, t_2 \in T_{\text{ground}}^\lambda(\Sigma_I)$, 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 $\oplus : \prod \alpha. \alpha \rightarrow \alpha \rightarrow \alpha$. We will omit its type argument. It is equipped with two reduction rules: $\oplus t s \rightarrow t$ and $\oplus t s \rightarrow s$. 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_{CPO} [9] guarantees that

- $\oplus t s \succ_{\text{CPO}} s$ by applying rule $@\triangleright$;
- $\oplus t s \succ_{\text{CPO}} t$ by applying rule $@\triangleright$ twice;
- $(\lambda t) s \succ_{\text{CPO}} t\{0 \mapsto s\}$ by applying rule $@\beta$.

Since this order is moreover monotone, it decreases with $\beta\oplus$ -reduction steps. The order is also well founded; thus, $\beta\oplus$ -reductions terminate. And since the $\beta\oplus$ -reduction steps describe

a finitely branching term 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.

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

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

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

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 $f\langle\bar{\tau}\rangle\bar{t}$ 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 $s\sigma = (f\langle\bar{\tau}\rangle\bar{t})\sigma$. 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 $\mathcal{F}((f\langle\bar{\tau}\rangle\bar{t})\theta) = f_m^{\bar{\tau}} \mathcal{F}(\bar{t}\theta)$ 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 λu 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 $s\theta \text{ diff}_{s\theta,s\theta'} \sim s\theta' \text{ diff}_{s\theta,s\theta'}$. Via β -conversion, this is equivalent to $v\theta \sim v\theta'$, where $v = u\{0 \mapsto \text{diff}_{s\theta,s\theta'}\}$. 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 $\text{diff}_{s\theta,s\theta'}$ 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 $s\sigma \rightarrow_{\beta} s'\sigma$ 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 $s\theta \sim s\theta'$. \square

Using the term rewrite system R , we define a higher-order interpretation $\mathcal{J}^{\text{IG}} = (\mathcal{U}^{\text{IG}}, \mathcal{J}^{\text{IG}}, \mathcal{D}^{\text{IG}}, \mathcal{L}^{\text{IG}})$. 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}_τ 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}_τ for each ground type τ and then let \mathcal{U}^{IG} be the set of all these domains \mathcal{D}_τ . We cannot identify the domains \mathcal{D}_τ with the first-order domains \mathcal{U}_τ because domains \mathcal{D}_τ for functional types τ must contain functions. Instead, we will define suitable domains \mathcal{D}_τ and a bijection \mathcal{E}_τ between \mathcal{U}_τ and \mathcal{D}_τ for each ground type τ .

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

$$\begin{aligned} \mathcal{D}_{\tau \rightarrow v} &= \{\varphi : \mathcal{D}_\tau \rightarrow \mathcal{D}_v \mid \exists s : \tau \rightarrow v. \forall u : \tau. \varphi(\mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R)) = \mathcal{E}_v([\![\mathcal{F}(s u)]\!]_R)\} \\ \mathcal{E}_{\tau \rightarrow v} &: \mathcal{U}_{\tau \rightarrow v} \rightarrow \mathcal{D}_{\tau \rightarrow v} \\ \mathcal{E}_{\tau \rightarrow v}([\![\mathcal{F}(s)]\!]_R)(\mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R)) &= \mathcal{E}_v([\![\mathcal{F}(s u)]\!]_R) \end{aligned}$$

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

- every element of $\mathcal{U}_{\tau \rightarrow v}$ is of the form $[\![\mathcal{F}(s)]\!]_R$ for some $s \in \mathcal{T}_{\text{ground}}(\Sigma_I)$;
- every element of \mathcal{D}_τ is of the form $\mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R)$ for some $u \in \mathcal{T}_{\text{ground}}(\Sigma_I)$;
- the definition does not depend on the choice of such s and u ; and
- $\mathcal{E}_{\tau \rightarrow v}([\![\mathcal{F}(s)]\!]_R) \in \mathcal{D}_{\tau \rightarrow v}$ for all $s \in \mathcal{T}_{\text{ground}}(\Sigma_I)$.

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}_τ are bijections. To prove the third claim, we assume that there are other terms $t \in \mathcal{T}_{\text{ground}}(\Sigma_I)$ and $v \in \mathcal{T}_{\text{ground}}(\Sigma_I)$ 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}_τ 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 $\mathcal{T}_{\text{ground}}(\Sigma_I)$, 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 $\mathcal{E}_{\tau \rightarrow v}$ above does not depend on the choice of s and u . The fourth claim is obvious from the definition of $\mathcal{D}_{\tau \rightarrow v}$ and the third claim.

It remains to show that $\mathcal{E}_{\tau \rightarrow v}$ is bijective. For injectivity, we fix two terms $s, t \in \mathcal{T}_{\text{ground}}(\Sigma_I)$ such that for all $u \in \mathcal{T}_{\text{ground}}(\Sigma_I)$, 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 $\mathcal{E}_{\tau \rightarrow v}$ is injective. For surjectivity, we fix an element $\varphi \in \mathcal{D}_{\tau \rightarrow v}$. By definition of $\mathcal{D}_{\tau \rightarrow v}$, there exists a term s such that $\varphi(\mathcal{E}_\tau([\![\mathcal{F}(u)]\!]_R)) = \mathcal{E}_v([\![\mathcal{F}(s u)]\!]_R)$ for all u . Hence, $\mathcal{E}_{\tau \rightarrow v}([\![\mathcal{F}(s)]\!]_R) = \varphi$, proving surjectivity and therefore bijectivity of $\mathcal{E}_{\tau \rightarrow v}$. Below, we will usually write \mathcal{E} instead of \mathcal{E}_τ since the type τ is determined by \mathcal{E}_τ '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 $[\![\perp]\!]$ and 1 with $[\![\top]\!]$. Moreover, we define $\mathcal{J}_{\text{ty}}^{\text{IG}}(\kappa)(\mathcal{D}_{\bar{\tau}}) = \mathcal{D}_{\kappa(\bar{\tau})}$ for all $\kappa \in \Sigma_{\text{ty}}$, completing the type interpretation of $\mathcal{J}_{\text{ty}}^{\text{IG}} = (\mathcal{U}^{\text{IG}}, \mathcal{J}_{\text{ty}}^{\text{IG}})$ and ensuring that $\mathcal{J}_{\text{ty}}^{\text{IG}}(o) = \mathcal{D}_o = \{0, 1\}$.

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

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

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

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

(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}^{IG} is an interpretation function.

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

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

Let $\mathcal{L}^{\text{IG}}(\xi, \lambda t) = \mathcal{E}(\llbracket \mathcal{F}((\lambda t)\theta) \rrbracket_R)$. 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 $\mathcal{D}_{\alpha\theta'} = \xi_{\text{ty}}(\alpha)$ for all α and $\mathcal{E}(\llbracket \mathcal{F}(x\theta') \rrbracket_R) = \xi_{\text{te}}(x)$ 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 $\llbracket \mathcal{F}(x\theta) \rrbracket_R = \llbracket \mathcal{F}(x\theta') \rrbracket_R$ for all x because \mathcal{E} is injective. By Lemma 4.42 it follows that $\llbracket \mathcal{F}((\lambda t)\theta) \rrbracket_R = \llbracket \mathcal{F}((\lambda t)\theta') \rrbracket_R$, which proves that \mathcal{L}^{IG} is well defined. This concludes the definition of the interpretation $\mathcal{J}^{\text{IG}} = (\mathcal{U}^{\text{IG}}, \mathcal{J}_{\text{ty}}^{\text{IG}}, \mathcal{J}^{\text{IG}}, \mathcal{L}^{\text{IG}})$. It remains to show that \mathcal{J}^{IG} is proper.

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

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

$$\llbracket t \rrbracket_{\mathcal{J}^{\text{IG}}} = \mathcal{E}(\llbracket \mathcal{F}(t \downarrow_\beta) \rrbracket_R)$$

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 $f\langle \bar{\tau} \rangle$, then

$$\begin{aligned} \llbracket t \rrbracket_{\mathcal{J}^{\text{IG}}} &= \mathcal{J}^{\text{IG}}(f, \mathcal{D}_{\bar{\tau}}) \\ &= \mathcal{E}(\llbracket \mathcal{F}(f\langle \bar{\tau} \rangle) \rrbracket_R) = \mathcal{E}(\llbracket \mathcal{F}(t \downarrow_\beta) \rrbracket_R) \end{aligned}$$

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

$$\begin{aligned} \llbracket t_1 t_2 \rrbracket_{\mathcal{J}^{\text{IG}}} &= \llbracket t_1 \rrbracket_{\mathcal{J}^{\text{IG}}}(\llbracket t_2 \rrbracket_{\mathcal{J}^{\text{IG}}}) \\ &\stackrel{\text{IH}}{=} \mathcal{E}_{\tau \rightarrow v}(\llbracket \mathcal{F}(t_1 \downarrow_\beta) \rrbracket_R)(\mathcal{E}_\tau(\llbracket \mathcal{F}(t_2 \downarrow_\beta) \rrbracket_R)) \\ &\stackrel{\text{Def } \mathcal{E}}{=} \mathcal{E}_v(\llbracket \mathcal{F}((t_1 t_2) \downarrow_\beta) \rrbracket_R) \end{aligned}$$

If t is a λ -expression, then

$$\begin{aligned} \llbracket \lambda u \rrbracket_{\mathcal{J}^{\text{IG}}}^\xi &= \mathcal{L}^{\text{IG}}(\xi, \lambda u) \\ &= \mathcal{E}(\llbracket \mathcal{F}((\lambda u)\theta \downarrow_\beta) \rrbracket_R) \\ &= \mathcal{E}(\llbracket \mathcal{F}((\lambda u) \downarrow_\beta) \rrbracket_R) \end{aligned}$$

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

We need to show that the interpretation \mathcal{J}^{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{J}^{IG} because we have not shown that \mathcal{J}^{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 $\xi'_{\text{ty}}(\alpha) = \llbracket \alpha\rho \rrbracket_{\mathcal{J}^{\text{IG}}}^{\xi_{\text{ty}}}$ for all type variables α and $\xi'_{\text{te}}(x) = \llbracket x\rho \rrbracket_{\mathcal{J}^{\text{IG}}}^\xi$ for all term variables x . We then have*

$$\llbracket t\rho \rrbracket_{\mathcal{J}^{\text{IG}}}^\xi = \llbracket t \rrbracket_{\mathcal{J}^{\text{IG}}}^{\xi'}$$

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:

$$\begin{aligned}
 \llbracket (\lambda u)\rho \rrbracket_{\mathcal{J}^{\text{IG}}}^\xi &= \llbracket \lambda u\rho \rrbracket_{\mathcal{J}^{\text{IG}}}^\xi \\
 &= \mathcal{L}^{\text{IG}}(\xi, \lambda u\rho) && \text{by the definition of the term denotation} \\
 &= \mathcal{E}(\llbracket \mathcal{F}((\lambda u)\rho\theta\downarrow_\beta) \rrbracket_R) && \text{for some } \theta \text{ by the definition of } \mathcal{L}^{\text{IG}} \\
 &= \mathcal{E}(\llbracket \mathcal{F}((\lambda u)\rho\downarrow_\beta) \rrbracket_R) && \text{because } (\lambda u)\rho \text{ is ground} \\
 &\stackrel{*}{=} \mathcal{L}^{\text{IG}}(\xi', \lambda u) && \text{by the definition of } \mathcal{L}^{\text{IG}} \text{ and Lemma 4.43} \\
 &= \llbracket \lambda u \rrbracket_{\mathcal{J}^{\text{IG}}}^{\xi'} && \text{by the definition of the term denotation}
 \end{aligned}$$

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

Lemma 4.45. *The interpretation \mathcal{J}^{IG} is proper.*

Proof. We need to show that $\llbracket \lambda t \rrbracket_{\mathcal{J}^{\text{IG}}}^{(\xi_{\text{ty}}, \xi_{\text{te}})}(a) = \llbracket t\{0 \mapsto x\} \rrbracket_{\mathcal{J}^{\text{IG}}}^{(\xi_{\text{ty}}, \xi_{\text{te}}[x \mapsto a])}$, where x is a fresh variable.

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

\square

Lemma 4.46. *\mathcal{J}^{IG} is term-generated; i.e., for all $\mathcal{D} \in \mathcal{U}^{\text{IG}}$ and all $a \in \mathcal{D}$, there exists a ground type τ such that $\llbracket \tau \rrbracket_{\mathcal{J}^{\text{IG}}_{\text{ty}}} = \mathcal{D}$ and a ground term t such that $\llbracket t \rrbracket_{\mathcal{J}^{\text{IG}}} = a$.*

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 $s \in \mathcal{T}_F$ such that $\llbracket s \rrbracket_R = \mathcal{E}^{-1}(a)$. Let $t = \mathcal{F}^{-1}(s)$. Then, by Lemma 4.43, $\llbracket t \rrbracket_{\mathcal{J}^{\text{IG}}} = \mathcal{E}(\llbracket s \rrbracket_R) = a$. \square

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

Proof. By Lemma 4.43, we have

$$\llbracket t \rrbracket_{\mathcal{J}^{\text{IG}}} = \mathcal{E}(\llbracket \mathcal{F}(t\downarrow_\beta) \rrbracket_R)$$

for any $t \in \mathcal{T}_{\text{ground}}(\Sigma_I)$. Since \mathcal{E} is a bijection, it follows that a ground literal $s \approx t$ in a clause $C \in \mathcal{C}_{\text{IG}}$ is true in \mathcal{J}^{IG} if and only if $\mathcal{F}(s \approx t)$ is true in R . So any closure $C \in \mathcal{C}_{\text{IG}}$ is true in \mathcal{J}^{IG} if and only if $\mathcal{F}(C)$ is true in R . \square

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

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. \square

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

It is inconvenient to construct a model of N_0 for the G level because \mathcal{I} converts parameters into subscripts. For example, in the model constructed in the previous subsubsection, it can happen that $a \approx b$ holds, but $f_a \approx f_b$ 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 $\mathcal{I}(N) \subseteq \mathcal{C}_{\text{IG}}$ instead of $N \subseteq \mathcal{C}_G$. 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 $N \subseteq \mathcal{C}_G$ be saturated up to redundancy w.r.t. $GRed_I$, and $\perp \notin N$. Then $\mathcal{I}^{\text{IG}} \models \mathcal{I}(N)$.*

Proof. This follows from Theorem 4.48 and Lemma 4.23. \square

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 h_{sel} be a selection function on \mathcal{C}_H (Definition 2.14).

Definition 4.50. A *derivation* is a finite or infinite sequence of sets $(N_i)_{i \geq 0}$ such that $N_i \setminus N_{i+1} \subseteq HRed_C(N_{i+1})$ 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 $\bigcup_i HRed_I(N_i)$.

Lemma 4.51. *The redundancy criteria $HRed_C$ and $HRed_I$ fulfill the following properties, as stated by Waldmann et al. [20]:*

- (R2) *if $N \subseteq N'$, then $HRed_C(N) \subseteq HRed_C(N')$ and $HRed_I(N) \subseteq HRed_I(N')$;*
- (R3) *if $N' \subseteq HRed_C(N)$, then $HRed_C(N) \subseteq HRed_C(N \setminus N')$ and $HRed_I(N) \subseteq HRed_I(N \setminus N')$;*
- (R4) *if $\iota \in HInf$ and $\text{concl}(\iota) \in N$, then $\iota \in HRed_I(N)$.*

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 $C \in \mathcal{C}_H$ and a grounding substitution θ , written $C \cdot \theta \blacktriangleright D \cdot \rho$, iff

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

Clearly, for all $C \in \mathcal{C}_H$ and all $N \subseteq \mathcal{C}_H$, we have $C \in HRed_C(N)$ if and only if for all grounding substitutions θ , we have

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

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

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

Since $C \in HRed_C(N)$, we know that

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

So it suffices to show that

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

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

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

Our induction hypothesis states:

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

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 $N' \subseteq HRed_C(N)$, and thus we have

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

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

(R3) for inferences:

Inspecting this definition of $HRed_I$ (Definition 2.25), we observe that to show that $HRed_I(N) \subseteq HRed_I(N \setminus N')$, it suffices to prove that

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

(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 $HRed_I$. We can equivalently write this as

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

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

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

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 $N' \subseteq HRed_C(N)$, and thus $N' \subseteq HRed_C(N \setminus N')$ by (R3) for clauses. So we have

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

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

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

Since $\text{concl}(\iota) \in N$ and $\text{concl}(\iota) = C_{m+1}$, 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 $\mathcal{F}(C_{m+1}\theta_{m+1}) \models_{\text{o}\lambda} \mathcal{F}(C_m\theta_m)$. 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, $\succ_{\mathcal{F}} = \succ_{\mathcal{F}}$. 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 $t' = \mathbf{T}$, 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 $t \approx \perp$, 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 $t \approx \perp$, then that literal is larger than $t \approx t'$ because if t' is Boolean, then $t' = \mathbf{T}$. 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 (O3)_F and (O4)_F. Thus the premise is larger than the conclusion.

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

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

For FARGCONG, the premise is larger than the conclusion by (O5)_F.

For FEXT, we use the condition that $u \succ_{\mathcal{F}} v$ and (O3)_F to show that $C[\mathcal{F}(v)]$ is smaller than the premise. We use $u \succ_{\mathcal{F}} v$ and (O5)_F to show that $\mathcal{F}(u \text{ diff} \langle \tau, v \rangle(u, v)) \not\approx \mathcal{F}(v \text{ diff} \langle \tau, v \rangle(u, v))$ is smaller than the premise. \square

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 $\perp \in N_i$ 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 $HRed_I$. By Lemma 4.29, $\mathcal{G}(N_\infty)$ is saturated up to redundancy w.r.t. $GInf$ and $GRed_I$.

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

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

Now \mathcal{J}^{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{J}^{IG} is term-generated by Lemma 4.46, there exists a substitution θ such

that $\llbracket \alpha\theta \rrbracket_{\mathcal{J}^{\text{IG}}} = \xi_{\text{ty}}(\alpha)$ for all type variables α in C and $\llbracket x\theta \rrbracket_{\mathcal{J}^{\text{IG}}} = \xi_{\text{te}}(x)$ 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 $\mathcal{J}^{\text{IG}} \models C\theta$. By Lemma 3.2, it follows that C is true w.r.t. ξ and \mathcal{J}^{IG} . Since ξ and $C \in N_0$ were arbitrary, we have $\mathcal{J}^{\text{IG}} \models N_0$. This contradicts condition 1 of the present theorem. \square

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

We define $\mathcal{J}'(\text{diff}, \mathcal{D}_1, \mathcal{D}_2, a, b)$ 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{J}' 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 $\text{diff}(\tau, v)(u, v)$ in t with a ground term s that does not contain `diff` such that $\llbracket s \rrbracket_{\mathcal{J}} = \mathcal{J}'(\text{diff}, \llbracket \tau \rrbracket_{\mathcal{J}_{\text{ty}}}^{\xi_{\text{ty}}}, \llbracket v \rrbracket_{\mathcal{J}_{\text{ty}}}^{\xi_{\text{ty}}}, \llbracket u \rrbracket_{\mathcal{J}}^{\xi}, \llbracket v \rrbracket_{\mathcal{J}}^{\xi})$. Such a term s exists because \mathcal{J} 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{J}' is a proper interpretation.

Since N does not contain `diff` and \mathcal{J} is a model of N , it follows that \mathcal{J}' is a model of N as well. \square

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

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

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

Proof. By Theorem 4.52 and Lemma 4.53. \square

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 manuscript). 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 König. 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.