
OPTIMISTIC LAMBDA-SUPERPOSITION

ALEXANDER BENTKAMP ^a, JASMIN BLANCHETTE ^a, MATTHIAS HETZENBERGER ^b,
AND UWE WALDMANN ^c

^a Ludwig-Maximilians-Universität München, Geschwister-Scholl-Platz 1, 80539 München, Germany
e-mail address: a.bentkamp@ifi.lmu.de, jasmin.blanchette@ifi.lmu.de

^b TU Wien Informatics, Favoritenstraße 9–11, 1040 Vienna, Austria
e-mail address: matthias.hetzenberger@tuwien.ac.at

^c Max Planck Institute for Informatics, Campus E1 4, 66123 Saarbrücken, Germany
e-mail address: uwe@mpi-inf.mpg.de

ABSTRACT. The λ -superposition calculus is a successful approach to proving higher-order formulas. However, some parts of the calculus are extremely explosive, notably due to the higher-order unifier enumeration and the functional extensionality axiom. In the present work, we introduce an “optimistic” version of λ -superposition that addresses these two issues. Specifically, our new calculus delays explosive unification problems using constraints stored along with the clauses, and it applies functional extensionality in a more targeted way. The calculus is sound and refutationally complete with respect to a Henkin semantics. We have yet to implement it in a prover, but examples suggest that it will outperform, or at least usefully complement, the original λ -superposition calculus.

1. INTRODUCTION

The (Boolean) λ -superposition calculus [6], which generalizes Bachmair and Ganzinger’s superposition calculus [1], has shown itself to be a powerful automated reasoning method for classical higher-order logic with function and Boolean extensionality. The calculus is sound and refutationally complete with respect to a Henkin semantics. It is implemented in the Zipperposition prover [30], and a refutationally incomplete, pragmatic version of it drives the E prover’s higher-order mode [32].

These implementations of λ -superposition achieve remarkable empirical results, but to do so, they must deprioritize or—in incomplete variants—disable specific features of the calculus that would otherwise cause combinatorial explosion. Among these features, the most problematic are the following:

- the hugely expensive computation of unifiers of flex–flex pairs, which the calculus requires instead of allowing Huet’s preunification procedure;
- the functional extensionality axiom and its orientation according to the term order, which enforces a lot of wasteful extensionality reasoning unrelated to the actual proof goal; and
- the so-called fluid superposition rule, which simulates rewriting below applied variables and which causes lots of inferences that rarely lead to a successful proof.

In this article, we introduce the *optimistic λ -superposition* calculus (Section 3), which addresses the first two issues:

- For unification, our new calculus delays explosive unification problems using constraints stored along with the clauses.
- For functional extensionality, it introduces a targeted inference rule that works in tandem with tailored term orders, described in a companion article [4]. The new rule works by first assuming that two functions are equal and delays the proof that they are equal on all arguments until the assumption is found to be useful.

Both of these new features delay some work and can be considered “optimistic,” hence the calculus’s name.

As a pleasant side effect of the new functional extensionality rule, we can strengthen the redundancy criterion used to simplify clauses. Some inference rules of the original λ -superposition calculus are now simplification rules in our new calculus.

Example 1.1. As an illustration of the stronger redundancy criterion, consider a derivation starting from the following clauses:

$$\begin{aligned} (1) \quad & (\lambda x. x + 1) \not\approx (\lambda x. 1 + x) \\ (2) \quad & y + z \approx z + y \end{aligned}$$

A negative extensionality inference from (1) yields the clause

$$(3) \quad \text{diff}(\lambda x. x + 1, \lambda x. 1 + x) + 1 \not\approx 1 + \text{diff}(\lambda x. x + 1, \lambda x. 1 + x)$$

which eventually leads to a derivation of the empty clause using (2). The original λ -superposition calculus required us to keep clause (1) and perform further inferences with it, whereas our new calculus can immediately discard (1) when generating (3).

We prove our calculus sound (Section 4) and refutationally complete (Section 5). The completeness proof is structured as six levels, starting from superposition for a ground first-order logic and culminating with nonground higher-order logic with functional extensionality. The parts of the proof concerned with the constraints attached to clauses and with the new functional extensionality rule are inspired by basic superposition [3, 23]. The two new features make the proof rather complicated, but the calculus is simpler than the original λ -superposition calculus in many respects:

- The intricate notions of “deep occurrences” and “fluid terms” play no role in our calculus.
- We removed the support for inner clausification, which does not perform well in practice and complicates the original λ -superposition calculus. As an additional benefit, this enables us to select literals of the form $t \approx \perp$ (a claim made for the original λ -superposition calculus as well [6] but corrected later [5, 25]).
- Our calculus does not require the presence of Hilbert’s choice axiom.

In principle, these simplifications could be applied to the original λ -superposition calculus as well without adding unification constraints or the new functional extensionality rule.

Our calculus’s two main features are inspired by Vampire’s higher-order mode [11], which is currently the best performing higher-order prover in CASC [28, 29]. Like our calculus, Vampire delays unification problems and functional extensionality proofs. The mechanisms are slightly different, however, because Vampire stores delayed unification problems in negative literals instead of constraints, allowing inference rules to be applied to them, and it uses unification with abstraction for functional extensionality (which is also implemented in E [32, p. 13]) instead of an inference rule. The similarities to performant

provers, along with example problems we have studied, suggest that our calculus not only is refutationally complete but will also perform well empirically. For further related work, we refer to Bentkamp et al. [6].

2. 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 [18], the TPTP TH1 standard [20], and the logic underlying λ -superposition by Bentkamp et al. [6].

Departing from Bentkamp et al., in the present work, quantifiers are not supported and must always be encoded as $(\lambda x. t) \approx (\lambda x. \mathbf{T})$ and $(\lambda x. t) \not\approx (\lambda x. \mathbf{L})$. 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.

2.1. 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 $(f(a_1), \dots, f(a_n))$.

2.1.1. 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 o 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 $e : \tau$.

2.1.2. 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 $\Pi \bar{a}_m. \bar{\tau}_n \Rightarrow v$, where $\bar{\tau}_n$ and v are types and \bar{a}_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 $\approx, \not\approx : \Pi \alpha. \alpha \rightarrow \alpha \rightarrow o$. A type signature and a term signature form a *signature*.

Our syntax makes use of a locally nameless notation [13] using De Bruijn indices [14]. 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 $t = f\langle\bar{v}_m\rangle(\bar{u})\ v$ 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.

2.1.3. 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 .

2.1.4. 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 .

2.1.5. 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 .

2.1.6. Constraints. A constraint is a term pair, written as $s \equiv t$. A set of constraints $s_1 \equiv t_1, \dots, s_n \equiv t_n$ is *true* if s_i and t_i are syntactically equal for all i . A set of constraints S is *satisfiable* if there exists a substitution such that $S\theta$ is true. A *constrained clause* $C[S]$ is a pair of a clause C and a finite set of constraints S . We write C_{H} for the set of all constrained clauses. Similarly, a *constrained term* $t[S]$ is a pair of a term t and a finite set of constraints S . Terms and clauses are special cases of constrained terms and constrained clauses where the set of constraints is empty. Given $C[S] \in C_{\text{H}}$ and a grounding substitution θ such that $S\theta$ is true, we call $C\theta$ a *ground instance* of $C[S]$. We write $\text{Gnd}(C[S])$ for the set of all ground instances of a constrained clause $C[S]$.

2.2. Semantics. The semantics is essentially the same as in Bentkamp et al. [6], 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{J}_{ty} and a type valuation ξ_{ty} is recursively defined by $\llbracket \alpha \rrbracket_{\mathcal{J}_{\text{ty}}}^{\xi_{\text{ty}}} = \xi_{\text{ty}}(\alpha)$ and $\llbracket \kappa(\bar{\tau}) \rrbracket_{\mathcal{J}_{\text{ty}}}^{\xi_{\text{ty}}} = \mathcal{J}_{\text{ty}}(\kappa)(\llbracket \bar{\tau} \rrbracket_{\mathcal{J}_{\text{ty}}}^{\xi_{\text{ty}}})$.

Given a type interpretation \mathcal{J}_{ty} and a type valuation ξ_{ty} , a *term valuation* ξ_{te} assigns an element $\xi_{\text{te}}(x) \in \llbracket \tau \rrbracket_{\mathcal{J}_{\text{ty}}}^{\xi_{\text{ty}}}$ 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 $\bar{a} \in [\bar{\tau}]_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}$ a value $\mathcal{J}(f, \bar{\mathcal{D}}_m, \bar{a}) \in [v]_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}$, 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) $\mathcal{J}(\perp) = 0$ | (I6) $\mathcal{J}(\rightarrow)(a, b) = \max\{1 - a, b\}$ |
| (I3) $\mathcal{J}(\mathbf{\Lambda})(a, b) = \min\{a, b\}$ | (I7) $\mathcal{J}(\approx, \mathcal{D})(c, d) = 1$ if $c = d$ and 0 otherwise |
| (I4) $\mathcal{J}(\mathbf{V})(a, b) = \max\{a, b\}$ | (I8) $\mathcal{J}(\not\approx, \mathcal{D})(c, d) = 0$ if $c = d$ and 1 otherwise |

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

The comprehension principle states that every function designated by a λ -expression is contained in the corresponding domain. Loosely following Fitting [17, 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 $[\tau]_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}$. 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 $[x]_{\mathcal{J}}^\xi = \xi_{\text{te}}(x)$, $[\mathbf{f}(\bar{\tau})(\bar{s})]_{\mathcal{J}}^\xi = \mathcal{J}(f, [\bar{\tau}]_{\mathcal{I}_{\text{ty}}}^{\xi_{\text{ty}}}, [\bar{s}]_{\mathcal{J}}^\xi)$, $[s \ t]_{\mathcal{J}}^\xi = [s]_{\mathcal{J}}^\xi ([t]_{\mathcal{J}}^\xi)$, and $[\lambda \langle \tau \rangle t]_{\mathcal{J}}^\xi = \mathcal{L}(\xi, \lambda \langle \tau \rangle t)$. For ground λ -terms t , the denotation does not depend on the choice of the valuation ξ , which is why we sometimes write $[t]_{\mathcal{J}}$ for $[t]_{\mathcal{J}}^\xi$.

An interpretation \mathcal{J} is *proper* if $[\lambda \langle \tau \rangle t]_{\mathcal{J}}^{(\xi_{\text{ty}}, \xi_{\text{te}})}(a) = [t \{ 0 \mapsto x \}]_{\mathcal{J}}^{(\xi_{\text{ty}}, \xi_{\text{te}}[x \mapsto a])}$ 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 $[s]_{\mathcal{J}}^\xi$ and $[t]_{\mathcal{J}}^\xi$ are equal (resp. different). A clause is true if at least one of its literals is true. A constrained clause $C[s_1 \equiv t_1, \dots, s_n \equiv t_n]$ is *true* if $C \vee s_1 \not\approx t_1 \vee \dots \vee s_n \not\approx t_n$ is true. A set of constrained clauses is true if all its elements are true. A proper interpretation \mathcal{J} is a *model* of a set N of constrained clauses, written $\mathcal{J} \models N$, if N is true in \mathcal{J} for all valuations ξ . Given two sets M, N of constrained clauses, we say that M *entails* N , written $M \models N$, if every model of M is also a model of N .

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

Definition 2.1. We call a proper interpretation \mathcal{J} *diff-aware* if \mathcal{J} is a model of the extensionality axiom—i.e.,

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

Given two sets M, N of constrained 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 .

3. CALCULUS

The optimistic λ -superposition calculus is designed to process an unsatisfiable set of higher-order clauses that have no constraints and do not contain constants with parameters, to enrich this clause set with clauses that may have constraints and may contain the constant `diff`, and to eventually derive an empty clause with satisfiable constraints.

Central notions used to define the calculus are *green subterms* (Section 3.1), which many of the calculus rules are restricted to, and *complete sets of unifiers up to constraints* (Section 3.2), which replace the first-order notion of a most general unifier. Existing unification algorithms must be adapted to cope with terms containing parameters (Section 3.3). The calculus is parameterized by a term order and a selection function, which must fulfill certain requirements (Section 3.4). The concrete term orders defined in a companion article fulfill the requirements (Section 3.5). Our core inference rules describe how the calculus derives new clauses (Section 3.6), and a redundancy criterion defines abstractly under which circumstances clauses may be deleted and when inferences may be omitted (Section 3.7). The abstract redundancy criterion supports a wide collection of concrete simplification rules (Section 3.8). Examples illustrate the calculus’s strengths and limitations (Section 3.9).

3.1. 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. [6] 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 3.1 ($\beta\eta$ -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 3.2 (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 $s, t \in T^{\text{pre}}(\Sigma)$, 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 3.3. 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 3.4. 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 3.5 (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 3.6. *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 $f(\bar{\tau})(\bar{u}) \bar{t}$ to $\lambda f(\bar{\tau})(\bar{u}) \bar{t} 0$. 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 3.7 (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 3.8. 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 3.9 (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 3.10. 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 $K.(f a).1$.

3.2. Complete Sets of Unifiers up to Constraints. Most of our calculus rules can be used in conjunction with Huet-style preunification, full unification, and various variants thereof. Only some rules require full unification. To formulate the calculus in full generality, we introduce the notion of a complete set of unifiers up to constraints. The definition closely resembles the definition of a complete set of unifiers, but allows us to unify only partially and specify the remainder in form of constraints.

Definition 3.11. 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 up to constraints* is a set P whose elements are pairs, each containing a substitution and a set of constraints, with the following properties:

- Soundness: For every $(\sigma, T) \in P$ and unifier ρ of T , $\sigma\rho$ is a unifier of S .
- Completeness: For every unifier θ of S , there exists a pair $(\sigma, T) \in P$ and a unifier ρ of T 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 let $\text{CSU}_X^{\text{upto}}(S)$ denote an arbitrary complete set of unifiers upto constraints with the following properties. First, to avoid ill-typed terms, we require that for the substitutions in $\text{CSU}_X^{\text{upto}}(S)$ unify the types of equated terms in S . In practice, this is not a severe restriction because type unification always terminates. Second, we require that the substitutions σ in $\text{CSU}_X^{\text{upto}}(S)$ are idempotent on X —i.e., $x\sigma\sigma = x\sigma$ for all $x \in X$, which can always be achieved by renaming variables.

The set X will consist of the free variables of the clauses that the constraints S originate from and will be left implicit.

Example 3.12. For the constraint $y \mathbf{a} \equiv f(z \mathbf{b})$ and $X = \{y, z\}$, the set $\{(\sigma, \{w \mathbf{a} \equiv z \mathbf{b}\})\}$ with $\sigma = \{y \mapsto \lambda f(w 0)\}$ is a complete set of unifiers up to constraints. It is sound because for every unifier ρ of $w \mathbf{a} \equiv z \mathbf{b}$, the substitution $\sigma\rho$ is a unifier of $y \mathbf{a} \equiv f(z \mathbf{b})$ since $(y \mathbf{a} \equiv f(z \mathbf{b}))\sigma = (f(w \mathbf{a}) \equiv f(z \mathbf{b}))$. It is complete because, for every unifier θ of $y \mathbf{a} \equiv f(z \mathbf{b})$, the term $y\theta$ must be of the form $\lambda f t$ for some preterm t , and then the substitution $\rho = \{w \mapsto \lambda t, z \mapsto z\theta\}$ is a unifier of $w \mathbf{a} \equiv z \mathbf{b}$ and fulfills $x\sigma\rho = x\theta$ for $x \in \{y, z\}$.

Definition 3.13. 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.

Equivalently, we could define a complete set of unifiers as a set P of substitutions such that $\{(\sigma, \emptyset) \mid \sigma \in P\}$ is a complete set of unifiers up to constraints.

The definitions above require $x\sigma\rho = x\theta$ only for variables $x \in X$, not for other variables, because the substitutions should be allowed to use auxiliary variables. For instance, in Example 3.12 above, for most unifiers θ , it is impossible to find a suitable ρ that fulfills $x\sigma\rho = x\theta$ for all variables x , including $x = w$.

When choosing a strategy to compute complete sets of unifiers up to constraints, there is a trade-off between how much computation time is spent and how precisely the resulting substitutions instantiate variables. At one extreme, we can compute a complete set of unifiers, which instantiates variables as much as possible. At the other extreme, the set containing

only the identity substitution and the original set of constraints is always a complete set of unifiers up to constraints, demonstrating that there exist terminating procedures that compute complete sets of unifiers up to constraints. In between these extremes lies Huet's preunification procedure [16, 19]. A good compromise in practice may be to run Huet's preunification procedure and to abort after a fixed number of steps, as described in the following subsection.

3.3. A Concrete Unification Strategy. As a strategy to compute CSU^{upto} , we suggest the following procedure, which is a bounded variant of Huet's preunification procedure, adapted to cope with polymorphism and parameters. This approach avoids coping with infinite streams of unifiers (except for rules that must use CSU instead of CSU^{upto}) and resembles Vampire's strategy [11].

Analogously to what we describe below, for CSU, one can extend procedures for the computation of complete sets of unifiers, such as Vukmirović et al.'s procedure [31], to cope with parameters.

Definition 3.14 (Flex-Flex, Flex-Rigid, Rigid-Rigid). Let $s \equiv t$ be a constraint. We write s and t in β -normal η -long form as $s = \lambda \cdots \lambda a u_1 \cdots u_p$ and $t = \lambda \cdots \lambda b v_1 \cdots v_q$, where a and b are variables, De Bruijn indices, or symbols (possibly with type arguments and parameters) and u_i and v_i are preterms. If a and b are both variables, we say that $s \equiv t$ is a flex-flex constraint. If only one of them is a variable, we say that $s \equiv t$ is a flex-rigid constraint. If neither a nor b is a variable, we say that $s \equiv t$ is a rigid-rigid constraint.

The Huet preunification procedure computes a substitution that unifies a set of constraints up to flex-flex pairs. It works as follows. Given a finite set of constraints S_0 , we construct a search tree whose nodes are either failure nodes \times or pairs (σ, S) of a substitution σ and a set S of constraints. The root node is the pair $(\{\}, S_0)$. Any node (σ, S) where S contains only flex-flex constraints is a successful leaf node. All failure nodes \times are also leaf nodes. To construct the children of any other node (σ, S) , we pick one of the constraints $s \equiv t \in S$ that is not a flex-flex constraint and apply the following rules:

- Type unification: We attempt to unify the types of s and t , which can be done using a first-order unification procedure. If the types are unifiable with a most general type unifier ρ , we add a child node $(\sigma\rho, S\rho)$. Otherwise, we add a child node \times .
- If the types of s and t are equal, we write $s = \lambda \cdots \lambda a u_1 \cdots u_p$ and $t = \lambda \cdots \lambda b v_1 \cdots v_q$ as in Definition 3.14 and apply the following rules:
 - Rigid-rigid cases: Let $S' = S \setminus \{s \equiv t\}$.
 - * If a and b are different De Bruijn indices, or if one of them is a symbol and the other a De Bruijn index, we add a child node \times .
 - * If a and b are identical De Bruijn indices, we add a child node $(\sigma, S' \cup \{u_1 \equiv v_1, \dots, u_p \equiv v_p\})$.
 - * If $a = f(\bar{s})(s_1, \dots, s_k)$ and $b = g(\bar{t})(t_1, \dots, t_l)$ with $f \neq g$, we add a child node \times .
 - * If $a = f(\bar{s})(s_1, \dots, s_k)$ and $b = f(\bar{v})(t_1, \dots, t_k)$ where \bar{s} and \bar{v} are not unifiable, we add a child node \times .
 - * If $a = f(\bar{s})(s_1, \dots, s_k)$ and $b = f(\bar{v})(t_1, \dots, t_k)$ where \bar{s} and \bar{v} are unifiable with a most general type unifier ρ , we add a child node $(\sigma\rho, (S' \cup \{s_1 \equiv t_1, \dots, s_k \equiv t_k, u_1 \equiv v_1, \dots, u_p \equiv v_p\})\rho)$.
 - Flex-rigid cases: Let $\tau_1 \rightarrow \dots \rightarrow \tau_p \rightarrow \tau$ be the type of a and $v_1 \rightarrow \dots \rightarrow v_q \rightarrow \tau$ be the type of b .

- * Imitation: If a is a variable x and b is either a De Bruijn index or a symbol (possibly with type arguments and parameters), we add a child node $(\sigma\rho, S\rho)$ with $\rho = \{x \mapsto \lambda\langle\tau_1\rangle \cdots \lambda\langle\tau_p\rangle b (y_1(p-1) \cdots 0) \cdots (y_q(p-1) \cdots 0)\}$, where y_1, \dots, y_q are fresh variables with y_i of type $\tau_1 \rightarrow \cdots \rightarrow \tau_p \rightarrow v_i$ for each i .
- * Projection: If a is a variable x and b is either a De Bruijn index or a symbol (possibly with type arguments and parameters), then for each $0 \leq i < p$ where $\tau_i = \tau'_1 \rightarrow \cdots \rightarrow \tau'_k \rightarrow \tau$ for some τ'_1, \dots, τ'_k , we add a child node $(\sigma\rho, S\rho)$ with $\rho = \{x \mapsto \lambda\langle\tau_1\rangle \cdots \lambda\langle\tau_p\rangle i (y_1(p-1) \cdots 0) \cdots (y_k(p-1) \cdots 0)\}$, where y_1, \dots, y_k are fresh variables with y_i of type $\tau_1 \rightarrow \cdots \rightarrow \tau_p \rightarrow \tau'_j$ for each j .
- * The same applies with the roles of a and b swapped.

Ultimately, the tree's leaf nodes are either failure nodes \mathbf{x} or success nodes (σ, S) , where S contains only flex-flex constraints and σ is the corresponding preunifier. Collecting all the preunifiers in the leaves yields the result of the standard, i.e., unbounded, Huet preunification procedure.

We propose to use a bounded variant instead to ensure that unification always terminates. In the bounded version, we construct the tree only up to a predetermined depth. Collecting all unifiers and their associated constraints in the leaves of the resulting tree also yields a complete set of unifiers up to constraints, which we can use in the role of the CSU^{upto} function of our core inference rules.

In addition, following Vukmirović et al. [31], we propose to extend this procedure with algorithms for decidable fragments such as pattern unification [22], fixpoint unification [19], and solid unification [31]. When one of these fragments applies to one of the constraints $s \equiv t$ of a node (σ, S) , the most general unifier ρ for this constraint can be determined in finite time, and we can add a single child node $(\sigma\rho, (S \setminus \{s \equiv t\})\rho)$ instead of the child nodes that would be added by the standard procedure.

Lemma 3.15. *The above procedure yields a complete set of unifiers up to constraints (Definition 3.11).*

Proof. Let S_0 be a set of constraints. Consider a search tree constructed by the above procedure. We must show that the successful leaves P of the tree form a complete set of unifiers up to constraints, i.e., we must show:

- Soundness: For every $(\sigma, T) \in P$ and unifier ρ of T , $\sigma\rho$ is a unifier of S_0 .
- Completeness: For every unifier θ of S_0 , there exists a pair $(\sigma, T) \in P$ and a unifier ρ of T such that $x\sigma\rho = x\theta$ for all $x \in X$.

For soundness, we prove the following more general property: For every node (σ, T) of the tree and every unifier ρ of T , the substitution $\sigma\rho$ is a unifier of S_0 . It is easy to check that the initial node has this property and that for each of the rules above, the constructed child node has the property if the parent node has it. Thus, soundness follows by induction on the structure of the tree.

For completeness, we prove the following more general property: Given a node (σ_0, U_0) of the tree and a unifier θ_0 of U_0 , there exists a pair $(\omega, V) \in P$ and a unifier π of V such that $x\omega\pi = x\sigma_0\theta_0$ for all $x \in X$.

Since the search tree is clearly finite, we can apply structural induction on the tree. So we may assume that the property holds for all child nodes of (σ_0, U_0) . We proceed by a case distinction analogous to the cases describing the procedure above.

If no decidable fragment applies to U_0 and U_0 contains only flex-flex pairs or if the depth limit has been reached, then (σ_0, U_0) is a leaf node. Then $(\sigma_0, U_0) \in P$ and the property holds with $\pi = \theta_0$.

Otherwise, if a decidable fragment applies to a constraint $s \equiv t \in U_0$ and provides a most general unifier ρ , then we have a child node $(\sigma_0\rho, (U_0 \setminus \{s \equiv t\})\rho)$. Since ρ is a most general unifier, there exists a substitution θ_1 such that $y\theta_0 = y\rho\theta_1$ for all variables y in $x\sigma_0$ with $x \in X$ and for all variables y in U_0 . So θ_1 is a unifier of $(U_0 \setminus \{s \equiv t\})\rho$ and by the induction hypothesis, there exists a pair $(\omega, V) \in P$ and a unifier π of V such that $x\omega\pi = x\sigma_0\rho\theta_1$ for all $x \in X$. Thus, $x\omega\pi = x\sigma_0\rho\theta_1 = x\sigma_0\theta_0$ for all $x \in X$, as required.

Otherwise, no decidable fragment applies to U_0 and U_0 contains a pair that is not flex-flex. Then our procedure picks one such pair $s \equiv t \in U_0$.

If the types of s and t are not equal, then they must be unifiable because θ_0 is a unifier. So there exists a child node $(\sigma_0\rho, U_0\rho)$, where ρ is the most general type unifier of s and t . We can then proceed as in the decidable fragment case above.

Otherwise, the types of s and t are equal. Let $s = \lambda \cdots \lambda a u_1 \cdots u_p$ and $t = \lambda \cdots \lambda b v_1 \cdots v_q$ as in Definition 3.14.

If $s \equiv t$ is a rigid-rigid pair, then a and b must be unifiable because θ_0 is a unifier. So a and b are either identical De Bruijn indices or unifiable symbols. In both cases, we can then proceed analogously to the decidable fragment case above.

If $s \equiv t$ is a flex-rigid pair, we assume without loss of generality that a is a variable x . Since θ_0 is a unifier of s and t and parameters cannot contain free De Bruijn indices, the term $a\theta_0$ must be either of the form $\lambda \cdots \lambda b \bar{s}$ for some terms \bar{s} or of the form $\lambda \cdots \lambda i \bar{s}$ for some De Bruijn index i and terms \bar{s} . In the first case, we apply the induction hypothesis to the child node produced by the imitation rule, and in the second case, we apply the induction hypothesis to the child node produced by the projection rule. In both cases, given the substitution ρ used by the rule, it is easy to construct a substitution θ_1 such that $y\theta_0 = y\rho\theta_1$ for all relevant variables y . Then we can proceed as in the decidable fragment case above. \square

For an efficient implementation, it is important to $\beta\eta$ -normalize terms and apply substitutions lazily, similarly to the approach of Vukmirović et al. [31].

3.4. 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 $x \mathbf{b} \succeq x \mathbf{a}$, where $\mathbf{b} \succ \mathbf{a}$. However, contrary to the claims made for the original λ -superposition calculus, employing the nonstrict term order can lead to incompleteness [7], 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 [26].

Definition 3.16 (Admissible Term Order). A relation \succ on terms and on constrained 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: $t\ll T \gg \succ s\ll S \gg$ implies $t\theta \succ s\theta$ for all grounding substitutions θ such that $T\theta$ and $S\theta$ are true;
- (O8) stability under grounding substitutions for clauses: $D\ll T \gg \succ C\ll S \gg$ implies $D\theta \succ C\theta$ for all grounding substitutions θ such that $T\theta$ and $S\theta$ are true;
- (O9) transitivity on constrained literals: the relation \succ on constrained literals is transitive.
- (O10) for all terms t and s such that $t \succ s$ and all substitutions θ such that for all type variables α , the type $\alpha\theta$ is ground and such that for all variables x , all variables in $x\theta$ are nonfunctional, if $s\theta$ contains a variable outside of parameters, then $t\theta$ must also contain that variable outside of parameters.

Definition 3.17 (Maximality). Given a term order \succ , a literal K of a constrained clause $C\ll S \gg$ is *maximal* if for all $L \in C$ such that $L\ll S \gg \succeq K\ll S \gg$, we have $L\ll S \gg \preceq K\ll S \gg$. 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 3.18 (Literal Selection Function). A literal selection function is a mapping from each constrained 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 3.19 (Eligibility). A literal L is (*strictly*) *eligible* w.r.t. a substitution σ in $C\ll S \gg$ if it is selected in $C\ll S \gg$ or there are no selected literals in $C\ll S \gg$ and $L\sigma$ is (*strictly*) maximal in $(C\ll S \gg)\sigma$.

A green position $L.s.p$ of a clause $C\ll S \gg$ 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 $C\ll S \gg$); and L is of the form $s \dot{\approx} t \in C$ such that $(s\ll S \gg)\sigma \not\preceq (t\ll S \gg)\sigma$.

3.5. Concrete Term Orders. A companion article [4] defines two concrete term orders fulfilling the criteria of Definition 3.16: 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]. We extend the orders to constrained clauses by ignoring the constraints.

Theorem 3.20. Let $\succ_{\lambda kbo}$ denote the strict variant of λ KBO as defined in the companion article. The order is parameterized by a precedence relation $>$ on symbols, a function w assigning weights to symbols, a constant w_{db} defining the weight of De Bruijn indices, and a function k assigning argument coefficients to symbols. Assume that these parameters fulfill $w(\top) = w(\perp) = 1$, $w_{db} \geq w(\text{diff})$, $f > \perp > \top$ for all symbols $f \notin \{\top, \perp\}$, and $k(\text{diff}, i) = 1$ for every i . Using the extension to constrained 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, $u \succ_{g\lambda kbo} \perp \succ_{g\lambda kbo} T$ for all ground terms $u \notin \{T, \perp\}$, using our assumptions about the weight and precedence of 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 [15] of a multiset, it is easy to see that stability under substitutions for terms implies stability under substitutions for clauses. Since we ignore the constraints in the order, we also have stability under substitutions for constrained 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. Since we ignore the constraints in the order, it is also transitive on constrained literals.
- (O10) By Theorem 5.14 of the companion article.

□

Theorem 3.21. *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 $f > \perp > T$ for all symbols $f \notin \{T, \perp\}$, that $\perp \leq ws$, and that $\text{diff} \leq ws$. Using the extension to constrained 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, $u \succ_{g\lambda lpo} \perp \succ_{g\lambda lpo} T$ for all ground terms $u \notin \{T, \perp\}$, using our assumptions about the precedence of 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 [15] of a multiset, it is easy to see that stability under substitutions for terms implies stability under substitutions for clauses. Since

we ignore the constraints in the order, we also have stability under substitutions for constrained 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. Since we ignore the constraints in the order, it is also transitive on constrained literals.
- (O10) By Theorem 5.20 of the companion article.

□

3.6. The Core Inference Rules. The optimistic λ -superposition calculus consists of the following core inference rules, which a priori must be performed to guarantee refutational completeness. The calculus is parameterized by an admissible term order \succ and a selection function $hsel$. We denote this calculus as $HInf^{\succ, hsel}$ or just $HInf$.

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

Definition 3.22. An *inference* ι is a tuple $(C_1, C_2, \dots, C_{n+1})$ of constrained clauses, written

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

The constrained 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:

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

1. $(\sigma, U) \in \text{CSU}^{\text{upto}}(T, S, t \equiv u)$;
2. u is not a variable;
3. $u\sigma$ is nonfunctional;
4. $(t\llbracket T \rrbracket)\sigma \not\leq (t'\llbracket T \rrbracket)\sigma$;
5. the position of u is eligible in $C\llbracket S \rrbracket$ w.r.t. σ ;
6. $t \approx t'$ is strictly maximal in $D\llbracket T \rrbracket$ w.r.t. σ ;
7. there are no selected literals in $D\llbracket T \rrbracket$.

The rule FLUIDSUP simulates superposition below applied variables:

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

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

1. $\sigma \in \text{CSU}(z \ t \equiv u)$;
2. u is not a variable but is variable-headed;
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:

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

Side conditions for EQRES:

1. $(\sigma, U) \in \text{CSU}^{\text{upto}}(S, u \equiv u')$;
2. $u \not\approx u'$ is eligible in $C [S]$ w.r.t. σ .

Side conditions for EQFACT:

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

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.

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

with the following side conditions:

1. $\sigma \in \text{CSU}(s \equiv s', t \equiv t')$;
2. $s \approx t$ is strictly eligible in $C [S]$ w.r.t. σ ;
3. s is not a variable;
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:

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

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

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 not a variable and is neither \top nor \perp ;
3. the position of u is eligible in $C [S]$ w.r.t. σ ;

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

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

1. u is not a variable but is variable-headed;
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 $C [S]$ w.r.t. σ .

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

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

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

with the following side conditions:

1. $(\sigma, U) \in \text{CSU}^{\text{upto}}(S, s \equiv \perp, t \equiv \top)$;
2. $s \approx t$ is strictly eligible in $C [S]$ 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.

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

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 $C [S]$ w.r.t. σ ;
3. x is a fresh variable.

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

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 $C \llbracket S \rrbracket$ w.r.t. σ .

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

with the following side conditions:

1. u is not a variable but 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 $C \llbracket S \rrbracket$ 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 .

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

3.7. 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 $\text{fun}_t : \tau_1 \times \dots \times \tau_n \Rightarrow (\tau \rightarrow v)$ for each expression t obtained by replacing each outermost proper yellow subterm in a higher-order term of type $\tau \rightarrow v$ by a placeholder symbol \square , where τ_1, \dots, τ_n are the types of the replaced subterms in order of occurrence.

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

Definition 3.23. For ground terms t , we define \mathcal{F} recursively as follows: If t is functional, then let t' be the expression obtained by replacing each outermost proper yellow subterm in t by the placeholder symbol \square , and let $\mathcal{F}(t) = \text{fun}_{t'}(\mathcal{F}(\bar{s}_n))$, where \bar{s}_n are the replaced subterms in order of occurrence. 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.

Example 3.24. $\mathcal{F}(\lambda (f (\lambda 1) (\lambda (\lambda 0)))) = \text{fun}_{\lambda (f (\lambda 1) \square)}(\text{fun}_{\lambda \square}(\text{fun}_{\lambda 0}))$.

Remark 3.25. A simpler yet equivalent formulation of the redundancy criterion can be obtained by defining $\mathcal{F}(t) = \text{fun}_t$ for functional terms t , without using the \square symbol. The completeness proof, however, would become more complicated.

Lemma 3.26. *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, $\llbracket \top \rrbracket_{\mathcal{I}}$ and $\llbracket \perp \rrbracket_{\mathcal{I}}$, 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 $\mathcal{I} \models_{\text{fol}} \mathcal{F}(u \text{ diff } \langle \tau, v \rangle(s, t)) \approx \mathcal{F}(v \text{ diff } \langle \tau, v \rangle(s, t))$.

As another building block of our redundancy criterion, we introduce the notion of trust. As a motivating example, consider the clauses $b \not\approx a$ and $b \approx a$, where $b \succ a$. Clearly, the empty clause can be derived via a SUP and a EQRES inference. If we replace the clause $b \approx a$ with the logically equivalent clause $x \not\approx a \llbracket x \equiv b \rrbracket$, however, an empty clause with satisfiable constraints cannot be derived because SUP does not apply at variables. In this sense, the clause $b \not\approx a$ is more powerful than $x \not\approx a \llbracket x \equiv b \rrbracket$. Technically, the reason for this is that the calculus is only guaranteed to derive contradictions entailed by so-called variable-irreducible instances of clauses, and the instance of $x \not\approx a \llbracket x \equiv b \rrbracket$ that maps x to b is not variable-irreducible. Since variable-irreducibility cannot be computed in general, when we replace a clause with another, we use the notion of trust to ensure that for every variable-irreducible instance of the replaced clause, there exists a corresponding variable-irreducible instance of the replacing clause. Concretely, for any variable that occurs in a constraint or in a parameter of the replacing clause, there must exist a variable in a similar context in the replaced clause.

Definition 3.27 (Trust). Let $C\theta$ be a ground instance of $C[S] \in \mathcal{C}_H$ and $D\rho$ be a ground instance of $D[T] \in \mathcal{C}_H$. We say that the θ -instance of $C[S]$ *trusts* the ρ -instance of $D[T]$ if for each variable x in D ,

- (i) for every literal $L \in D$ containing x outside of parameters, there exists a literal $K \in C$ and a substitution σ such that $z\theta = z\sigma\rho$ for all variables z in C and $L \preceq K\sigma$; or
- (ii) x neither occurs in parameters in D nor appears in T .

The most general form of redundancy criteria for constrained superposition calculi are notoriously difficult to apply to concrete simplification rules. In the spirit of Nieuwenhuis and Rubio [23], we therefore introduce a simpler, less general notion of redundancy that suffices for most simplification rules. We provide a simple criterion for clauses and one for inference rules.

3.7.1. Simple 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 with constraints. 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 C_H , which prevents infinite chains of clauses where each clause is made redundant by the next one.

Definition 3.28 (Simple Clause Redundancy). Let $N \subseteq C_H$ and $C[S] \in C_H$. We call $C[S]$ *simply redundant* w.r.t. N , written $C[S] \in HRed_C^*(N)$, if for every $C\theta \in \text{Gnd}(C[S])$ at least one of the following two conditions holds:

1. There exist an indexing set I and for each $i \in I$ a ground instance $D_i\rho_i$ of a clause $D_i[T_i] \in N$, such that
 - (a) $\mathcal{F}(\{D_i\rho_i \mid i \in I\}) \models_{\text{o}\lambda} \mathcal{F}(C\theta)$;
 - (b) for all $i \in I$, $D_i\rho_i \prec C\theta$; and
 - (c) for all $i \in I$, the θ -instance of $C[S]$ trusts the ρ_i -instance of $D_i[T_i]$.
2. There exists a ground instance $D\rho$ of some $D[T] \in N$ such that
 - (a) $D\rho = C\theta$;
 - (b) $C[S] \sqsupseteq D[T]$; and
 - (c) the θ -instance of $C[S]$ trusts the ρ -instance of $D[T]$.

Remark 3.29. Although the calculus is refutationally complete for any choice of \sqsupseteq , we propose the following definition for \sqsupseteq . Given a clause $C[S]$ with nonempty S and a clause D with no constraints, we define $C[S] \sqsupseteq D$. For two clauses C and D with no constraints, following Bentkamp et al. [8, Sect. 3.4], we propose 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 .

3.7.2. Simple 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. For simplicity, there is no selection function, but our notion of eligibility is adapted to overapproximate any possible selection function as follows. A literal $L \in C$ is (*strictly*) *eligible* in C if L is negative or if L is of the form $t \approx \perp$ or if L is (*strictly*) maximal in C . A position $L.s.p$ of a clause C is *eligible* if the literal L is of the form $s \approx t$ with $s \succ t$ and L is either negative and eligible or positive and strictly eligible.

We define green subterms on first-order terms as follows. Every term is a green subterm of itself. Every direct subterm of a nonfunctional green subterm is also a green subterm. In keeping with our notation for higher-order terms, we write $t\langle u \rangle$ for a term t containing a green subterm u .

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

Side conditions for FSUP:

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

No side conditions for FEQRES. Side conditions for FEQFACT:

1. $u \approx v$ is maximal in C ;
2. $u \succ v$.

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:

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

Side conditions for FBOOLHOIST and FLOOBHOIST:

1. u is of Boolean type;
2. $u \neq \perp$ and $u \neq \top$;
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 $u \approx \top$.

Side condition 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 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 , and s are ground terms.

Definition 3.30. Since \mathcal{F} is bijective on ground terms by Lemma 3.26, 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 3.31. Let $\iota \in HInf^{\succ, hsel}$ for a term order \succ and a selection function $hsel$. Let $C_1[S_1], \dots, C_m[S_m]$ be its premises and $C_{m+1}[S_{m+1}]$ 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

– $S_1\theta_1, \dots, S_{m+1}\theta_{m+1}$ are true and

–

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

is a valid $FInf^{\succ_{\mathcal{F}}}$ inference ι' such that the rule names of ι and ι' correspond up to the prefixes F and FLUID.

Definition 3.32 (Simple Inference Redundancy). Let $N \subseteq \mathcal{C}_H$. Let $\iota \in HInf$ an inference with premises $C_1[S_1], \dots, C_m[S_m]$ and conclusion $C_{m+1}[S_{m+1}]$. We call ι *simply redundant* w.r.t. N , written $\iota \in HRed_I^*(N)$, if for every tuple of substitutions $(\theta_1, \dots, \theta_{m+1})$ for which ι is rooted in $FInf$ (Definition 3.31), there exists an index set I and for each $i \in I$ a ground instance $D_i\rho_i$ of a clause $D_i[T_i] \in N$ such that

1. $\mathcal{F}(\{D_i\rho_i \mid i \in I\}) \models_{o\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$;
2. ι is a DIFF inference or for all $i \in I$, $D_i\rho_i \prec C_m\theta_m$; and
3. for all $i \in I$, the θ_{m+1} -instance of $C_{m+1}[S_{m+1}]$ trusts the ρ_i -instance of $D_i[T_i]$.

3.8. Simplification Rules.

3.8.1. Analogues of First-Order Simplification Rules. Our notion of simple clause redundancy (Definition 3.28) can justify most analogues of the simplification rules implemented in Schulz's E prover [27, Sections 2.3.1 and 2.3.2]. Deletion of duplicated literals, deletion of resolved literals, and syntactic tautology deletion adhere to our redundancy criterion, even when the involved clauses carry constraints. Semantic tautology deletion can be applied as well, even on constrained clauses, but we must use the entailment relation $\models_{o\lambda}$ under the encoding \mathcal{F} . Positive and negative simplify-reflect can be applied as well, even with constraints, as long as the substitution makes each constraint of the unit clause true or translates it into a constraint already present on the other clause.

Our analogue of clause subsumption is the following. The subsumed clause can have constraints, but the subsuming clause cannot.

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

with the following side conditions:

1. $D \neq \perp$ or $C\sigma[S] \sqsupseteq C$;
2. C does not contain a variable occurring both inside and outside of parameters.

Lemma 3.33. SUBSUMPTION can be justified by simple clause redundancy.

Proof. Let $(C\sigma \vee D)\theta \in \text{Gnd}(C\sigma \vee D[S])$.

If D is nonempty, we apply condition 1 of Definition 3.28, using $I = \{\ast\}$, $D_* = C$ and $\rho_* = \sigma\theta$. The clause $C\sigma\theta$ is a proper subclause of $(C\sigma \vee D)\theta$ and therefore $\mathcal{F}(C\sigma\theta) \models_{o\lambda} \mathcal{F}((C\sigma \vee D)\theta)$ (condition 1a) and $C\sigma\theta \prec (C\sigma \vee D)\theta$ (condition 1b). For condition 1c, let x be a variable in C . By condition 2 of SUBSUMPTION, x occurs only inside parameters or only outside parameters. If it occurs only inside, we apply condition (i) of Definition 3.27; if it occurs only outside, we apply condition (ii) of Definition 3.27.

If D is \perp , we apply condition 2 of Definition 3.28, using C for D and $\sigma\theta$ for ρ . Condition 2a clearly holds. Condition 2b holds by condition 1 of SUBSUMPTION. Condition 2c follows from condition 2 of SUBSUMPTION as above. \square

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

Lemma 3.34. 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 3.35. 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 $u \llangle v \rrangle_p$ and $u \llangle v' \rrangle_p$ 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

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

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 3.34, we have $u \llangle v \rrangle_p = u \langle \lambda w_1 \langle \lambda w_2 \langle \dots w_n \langle v \rangle \dots \rangle \rangle \dots \gg$.

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

$$\mathcal{I} \models_{o\lambda} \mathcal{F}(u \llangle v \rrangle_p) \approx u \llangle v' \rrangle_p$$

\square

Our variant of rewriting of positive and negative literals (demodulation) is the following. The rewritten clause can have constraints, but the rewriting clause cannot.

$$\frac{t \approx t' \quad C \llangle v \rrangle [S]}{t \approx t' \quad C \llangle v' \rrangle [S]} \text{ DEMOD}$$

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

4. $t \approx t'$ does not contain a variable occurring both inside and outside of parameters.

Remark 3.36. In general, it is unclear how to compute condition 3 of DEMOD. For λ KBO and λ LPO described in Section 3.5, 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 $\text{diff}(_, _)(_, _)$. 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 3.5 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 3.5 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 3.37. DEMOD can be justified by simple clause redundancy, regardless of the choice of $\beta\eta$ -normalizer.

Proof. We apply condition 1 of Definition 3.28, using $C\ll v \gg [S]$ for $C[S]$. Let $C\ll v \gg \theta \in \text{Gnd}(C\ll v \gg [S])$. Let $*$ be a placeholder we use to extend a set of terms by an additional element. Then we set

$$\begin{aligned} I &= \{\bar{t}_k \mid \text{each } t_i \text{ is a ground term of the form } \text{diff}(_, _)(_, _) \} \cup \{*\} \\ D_{\bar{t}_k}[\![T_{\bar{t}_k}]\!] &= t \approx t' \\ \rho_{\bar{t}_k} &= \sigma\{\bar{x}_k \mapsto \bar{t}_k\}\theta \\ D_*[\![T_*]\!] &= C\ll v' \gg [S] \\ \rho_* &= \theta \end{aligned}$$

By condition 1 of DEMOD,

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

for each tuple \bar{t}_k , where each t_i is of the form $\text{diff}(_, _)(_, _)$.

By Lemma 3.35, $\mathcal{F}(\{D_i\rho_i \mid i \in I \setminus \{*\}\}) \models_{\text{o}\lambda} \mathcal{F}(u\theta\ll v\theta \gg \approx u\theta\ll v'\theta \gg)$, where u is a side of a literal in $C\ll v \gg$ containing the orange subterm v . Thus $\mathcal{F}(\{D_i\rho_i \mid i \in I\}) \models_{\text{o}\lambda} \mathcal{F}(C\ll v \gg \theta)$ (condition 1a of simple redundancy). Condition 2 and 3 of DEMOD imply $D_i\rho_i \prec C\ll v \gg \theta$ for all $i \in I$ (condition 1b of simple redundancy).

For condition 1c of simple redundancy, consider first a variable in $D_{\bar{t}_k} = t \approx t'$. By condition 4 of DEMOD, either condition (ii) or (if the variable occurs only inside parameters) (i) of trust is fulfilled. Second, consider a variable x in $C\ll v' \gg$. Then we apply condition (i) of trust. For every literal $L \in C\ll v' \gg$ that contains x outside of parameters, we use the corresponding literal $K \in C\ll v \gg$ and the identity substitution for the σ of condition (i). By condition 2, $L \preceq K$. \square

Our variant of equality subsumption is the following:

$$\frac{t \approx t' \quad C' \vee s \ll v \gg \approx s' \ll v' \gg [S]}{t \approx t'} \text{EQUALITYSUBSUMPTION}$$

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 $C \ll v \gg \succ v\{(0, \dots, k-1) \mapsto \bar{t}_k\} \approx v'\{(0, \dots, k-1) \mapsto \bar{t}_k\}$;
3. $t \approx t'$ does not contain a variable occurring both inside and outside of parameters.

To compute condition 2, we can exploit Remark 3.36.

Lemma 3.38. EQUALITYSUBSUMPTION can be justified by simple clause redundancy, regardless of the choice of $\beta\eta$ -normalizer.

Proof. Analogous to Lemma 3.37. \square

3.8.2. Additional Simplification Rules. The core inference rules ARGCONG, CLAUSIFY, FALSEELIM, LOOBHOIST, and BOOLHOIST described in Section 3.6 can under certain conditions be applied as simplification rules.

Lemma 3.39. ARGCONG can be justified as a simplification rule by simple clause redundancy when σ is the identity. Moreover, it can even be applied when its eligibility condition does not hold.

Proof. Let $C\theta$ be a ground instance of $C[S]$. Let $\tau \rightarrow v$ be the type of $s\theta$ and $s'\theta$. We apply condition 1 of Definition 3.28, using $I = \{(u, v) \mid u, v : \tau \rightarrow v \text{ ground}\}$, $D_{(u,v)} = C' \vee s x \approx s' x$, $T_{(u,v)} = S$, and $\rho_{(u,v)} = \theta[x \mapsto \text{diff}\langle\tau, v\rangle(u, v)]$. Condition 1a follows from the extensionality property of $\models_{o\lambda}$. Condition 1b follows from (O5).

For condition 1c, first consider the fresh variable x . Since x is fresh and parameters cannot contain free De Bruijn indices, x cannot occur in parameters in $C' \vee s x \approx s' x$, and thus condition (ii) of Definition 3.27 applies.

Now consider any other variable y in $D_{(u,v)}$. Such a variable must occur in C . We apply condition (i) of Definition 3.27, using the identity substitution for σ and—if y occurs in s or s' —using (O5). \square

Lemma 3.40. CLAUSIFY can be justified as a simplification rule by simple 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 3.28, using the fact that $\models_{o\lambda}$ interprets Booleans. \square

Lemma 3.41. FALSEELIM can be justified as a simplification rule by simple 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 3.28, using the fact that $\models_{o\lambda}$ interprets Booleans. \square

Lemma 3.42. BOOLHOIST and LOOBHOIST can be justified to be applied together as a simplification rule by simple 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 3.28, using the fact that $\models_{\text{o}\lambda}$ interprets Booleans. \square

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

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

Lemma 3.43. NOTTRUE and NOTFALSE can be justified as simplification rules by simple clause redundancy.

Proof. By condition 1 of Definition 3.28, using the fact that $\models_{\text{o}\lambda}$ interprets Booleans. \square

The following simplification rule, UNIF, allows us to run a unification procedure to remove the constraints of a clause.

$$\frac{C[S]}{C\sigma_1 \quad \dots \quad C\sigma_n} \text{UNIF}$$

with the following side conditions:

1. $\{\sigma_1, \dots, \sigma_n\}$ is a complete set of unifiers for S ;
2. $C[S] \sqsupseteq C\sigma_i$ for all i .

Lemma 3.44. UNIF can be justified by simple clause redundancy.

Proof. Let $C\theta \in \text{Gnd}(C[S])$. By condition 1 of UNIF and Definition 3.13, there must exist an index i and a substitution ρ such that $z\sigma_i\rho = z\theta$ for all z in $C[S]$. We apply condition 2 of Definition 3.28. We use $C\sigma_i$ for D and ρ for ρ . Condition 2a follows from the fact that $z\sigma_i\rho = z\theta$ for all z in $C[S]$. Condition 2b follows from condition 2 of UNIF.

For condition 2c, we must show that the θ -instance of $C[S]$ trusts the ρ -instance of $C\sigma_i$. We will use condition (i) of trust. Let $L \in C\sigma_i$. Let K be a literal in C such that $K\sigma_i = L$. We use σ_i for σ . Then we have $z\theta = z\sigma_i\rho = z\sigma\rho$ for all variables z in C . Moreover, $K\sigma = K\sigma_i = L$ implies $L \preceq K\sigma$. \square

The following rule is inspired by one of Leo-II's extensionality rules [9]:

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

Lemma 3.45. NEGEXT can be justified by simple clause redundancy.

Proof. Let $C\theta$ be a ground instance of $C[S]$. We apply condition 1 of Definition 3.28, using $I = \{*\}$, $D_* = C' \vee s \text{ diff}(\tau, v)(s, s') \not\approx s' \text{ diff}(\tau, v)(s, s')$, $T_* = S$, and $\rho_* = \theta$. Condition 1a follows from the argument congruence property of $\models_{\text{o}\lambda}$. Condition 1b follows from (O5). For condition 1c, consider a variable y in C . We apply condition (i) of Definition 3.27, using the identity substitution for σ and (O5). \square

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

Example 3.46 (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 [5, 25].

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 3.47 (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 3.48 (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 3.49 (Delaying Unification Using Constraints). Consider the following clause set:

- (1) $\text{map}(\lambda u. y(s u)) a \not\approx \text{map}(\lambda u. z(s(y u))) a \vee \text{lt}(y \text{zero})(s(s(s \text{zero}))) \approx \mathbf{T}$
- (2) $\text{lt } x x \approx \perp$

We assume that the first literal of (1) is selected. Using a CSU^{upto} function that implements Huet’s preunification procedure, we can derive the following clauses:

- (3) $\text{lt}(y \text{zero})(s(s(s \text{zero}))) \approx \mathbf{T} \llbracket \lambda u. y(s u) \equiv \lambda u. z(s(y u)) \rrbracket$ (by EQRES from (1))
- (4) $\perp \approx \mathbf{T}$ (by SUP from (3),(2))
- (5) \perp (by CLAUSIFY from (4))

If our calculus did not support constraints, we would have to solve the unification problem in (3) first, which yields an infinite number of solutions among which the simplest ones are dead ends.

Example 3.50 (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 3.39, 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 3.51 (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 3.52. This example illustrates why condition (ii) of our definition of trust (Definition 3.27) must require the variable not to occur in parameters. 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) $p \text{ diff}_{\lambda x. (\neg p x y) \wedge (p x y \vee y \not\simeq a), \lambda x. \perp} y \approx \perp$
- (7) $p \text{ diff}_{\lambda x. (\neg p x y) \wedge (p x y \vee y \not\simeq a), \lambda x. \perp} y \approx T \vee y \not\simeq a$
- (8) $p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \not\simeq a), \lambda x. \perp} b \approx \perp$
- (9) $p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \not\simeq a), \lambda x. \perp} b \approx T \vee b \not\simeq a$

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

$$(10) \quad p \text{ diff}_{\lambda x. (\neg p x b) \wedge (p x b \vee b \not\simeq a), \lambda x. \perp} a \approx T$$

In this derivation, (2), (3), (4), (5), and (9) can be deleted because `NEGEXT`, `NOTFALSE`, `CLAUSIFY`, `DEMOD`, and positive simplify-reflect can be applied as simplification rules.

To illustrate why condition (ii) does not apply to variables that occur in parameters, we also remove (8), which is against the redundancy criterion but would be justified by `SUBSUMPTION` of (8) by (6) if condition (ii) ignored parameters. The following clauses remain:

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

Assuming that the negative literal in (7) is selected and that $b \succ a$, no core inference rule other than `DIFF` applies. Due to the explosive nature of `DIFF`, it is difficult to predict whether `DIFF` inferences lead anywhere, but we conjecture that this is indeed a counterexample to a redundancy criterion that ignores parameters.

An alternative approach with a stronger redundancy criterion that does not need to treat parameters specially may be to enforce superposition inferences into variables that have other occurrences inside parameters. In the example above, this would entail a superposition inference from (1) into the variable y in the second literal of (7), which would indeed lead to a refutation.

4. SOUNDNESS

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

Lemma 4.1 (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 $\xi'_{\text{ty}}(\alpha) = \llbracket \alpha\theta \rrbracket_{\mathcal{J}_{\text{ty}}}^{\xi_{\text{ty}}}$ 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 $t = f\langle\bar{\tau}\rangle(\bar{u})$:

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

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

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

□

Lemma 4.2 (Substitution Lemma for Clauses). *Let θ be a substitution, and let C be a clause. For any proper interpretation $\mathcal{J} = (\mathcal{J}_{\text{ty}}, \mathcal{J}, \mathcal{L})$ and any valuation ξ , $C\theta$ is true w.r.t. \mathcal{J} and ξ if and only if C is true w.r.t. \mathcal{J} and ξ' , where the modified valuation ξ' is defined by $\xi'_{\text{ty}}(\alpha) = \llbracket \alpha\theta \rrbracket_{\mathcal{J}_{\text{ty}}}^{\xi_{\text{ty}}}$ 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{J} and ξ if and only if one of its literals is true w.r.t. \mathcal{J} 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{J} 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 4.1, $\llbracket s\theta \rrbracket_{\mathcal{J}}^{\xi}$ 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{J} and ξ' . This holds if and only if C is true w.r.t. \mathcal{J} and ξ' . □

Theorem 4.3. *All core inference rules are sound w.r.t. \approx (Definition 2.1). 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 4.2, \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. \square

5. REFUTATIONAL COMPLETENESS

Superposition is a saturation-based calculus. Provers that implement it start from an initial clause set N_0 and repeatedly add new clauses by performing inferences or remove clauses by determining them to be redundant. In the limit, this process results in a (possibly infinite) set N_∞ of persistent clauses. Assume that inferences are performed in a fair fashion; i.e., no nonredundant inference is postponed forever. Then the set N_∞ is saturated, meaning that all inferences are redundant (for example because their conclusion is in the set). Refutational completeness is the property that if N_∞ does not contain the empty clause, N_0 has a model. Since refutational completeness is the only kind of completeness that interests us in this article, we will also refer to it as “completeness.”

Due the role of constraints and parameters in our calculus, our completeness result, stated in Corollary 5.96, makes two additional assumptions: It assumes that the clauses in N_0 have no constraints and do not contain constants with parameters. And, instead of the usual assumption that N_∞ does not contain the empty clause, we assume that N_∞ does not contain an empty clause with satisfiable constraints.

5.1. Proof Outline. The idea of superposition completeness proofs in general is the following: We assume that N_∞ does not contain the empty clause. We construct a term rewrite system derived from the ground instances of N_∞ . We view this system as an interpretation \mathcal{I} and show that it is a model of the ground instances and thus of N_∞ itself. Since only redundant clauses are removed during saturation, \mathcal{I} must also be a model of N_0 .

Completeness proofs of *constrained* superposition calculi, including our the completeness proof of our calculus, must proceed differently. The constraints prevent us from showing \mathcal{I} to be a model of all ground instances of N_∞ . Instead, we restrict ourselves to proving that \mathcal{I} is a model of the variable-irreducible ground instances of N_∞ . Roughly speaking, a variable-irreducible ground instance is one where the terms used to instantiate variables are irreducible w.r.t. the constructed term rewrite system. The notion of redundancy must be based on the notion of variable-irreducible ground instances as well, so that if \mathcal{I} is a model of the variable-irreducible ground instances of N_∞ , it is also a model of the variable-irreducible

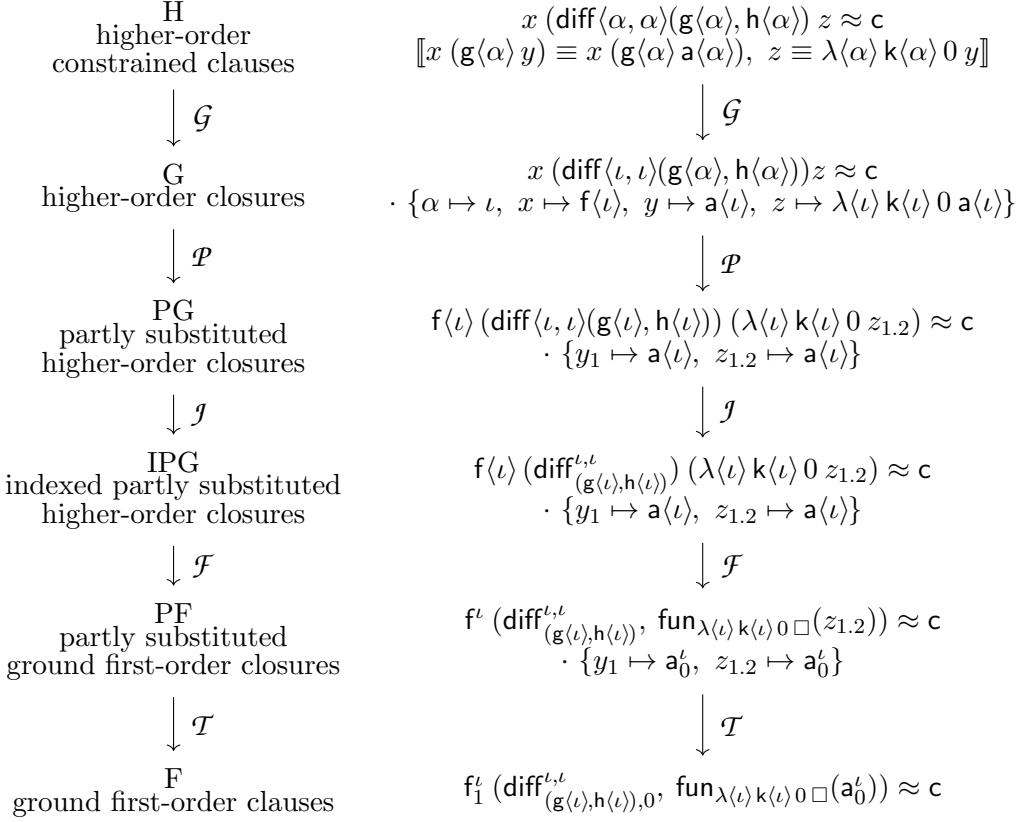


Figure 1: Overview of the levels

ground instances of N_0 . Assuming that the initial clauses N_0 do not have constraints, \mathcal{I} is then a model of all ground instances of N_0 because every ground instance has a corresponding variable-irreducible instance with the same truth value in \mathcal{I} . It follows that \mathcal{I} is a model of N_0 .

To separate concerns, our proof is structured as a sequence of six levels, most of which have their own logic, calculus, redundancy criterion, and completeness property. The levels are called H, G, PG, IPG, PF, and F. They are connected by functions encoding clauses from one level to the next.

The level H is the level of higher-order constrained clauses, using the logic described in Section 2 and the calculus described in Section 3. Our ultimate goal is to prove completeness on this level.

The level G is the level of higher-order closures, where a closure $C \cdot \theta$ is a pair consisting of a clause C and a grounding substitution θ . The function \mathcal{G} maps each clause from level H to a corresponding set of closures on level G using all possible grounding substitutions.

The level PG is the level of *partly substituted* higher-order closures. It is the fragment of G that contains no type variables and no functional variables. The map \mathcal{P} encodes closures from G into level PG by applying a carefully crafted substitution to functional variables.

The level IPG is the level of *indexed* partly substituted closures. It modifies the signature of the previous levels by replacing each symbol with parameters $f : \prod \bar{a}_m. \bar{\tau}_n \Rightarrow \tau$ by a

collection of symbols $f_{\bar{u}_n}^{\bar{v}_m} : \tau$ for each tuple of types \bar{v}_m and each tuple of ground terms $\bar{u}_n : \bar{\tau}_n$. The map \mathcal{J} encodes closures from PG into IPG by moving type arguments into the superscript indices \bar{v}_m parameters into the subscript indices \bar{u}_n .

The level PF is the level of partly substituted ground first-order closures. Its logic is the one described in Section 3.7 except with variables and closures. We extend the encoding \mathcal{F} (Definition 3.23) with variables, yielding an encoding from IPG into PF.

The level F is the level of first-order clauses. It uses the same logic as PF but uses ground clauses instead of closures. The map \mathcal{T} connects the two by mapping a closure $C \cdot \theta$ to the clause $C\theta$.

Figure 1 gives an overview of the hierarchy of levels and an example of a clause instance across the levels.

5.2. 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 3. 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 $f_{\bar{t}_n}^{\bar{v}_m} : \tau$, 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 5.1. In the subscripts t_i of constants $f_{\bar{t}_n}^{\bar{v}_m} \in \Sigma_I$, we identify each term of the form $f(\bar{v}_m)(\bar{t}_n) \in \mathcal{T}_{\text{ground}}(\Sigma_H)$ 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 3.7 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 5.2. In the subscripts of constants fun_t in $\mathcal{F}(\Sigma_H)$ and $\mathcal{F}(\Sigma_I)$, we identify each term of the form $f(\bar{v}_m)(\bar{t}_n) \in \mathcal{T}_{\text{ground}}(\Sigma_H)$ 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.

Our completeness proof uses two sets of variables. Let \mathcal{V}_H be the set of variables used by the calculus described in Section 3. Based on \mathcal{V}_H , we define the variables \mathcal{V}_{PG} of the PG level as

$$\mathcal{V}_{\text{PG}} = \mathcal{V}_H \cup \{y_p(\tau) \mid y \in \mathcal{V}_H, p \text{ a list of natural numbers}, \tau \text{ a nonfunctional type}\}$$

The table below summarizes our completeness proof's six levels, each with a set of terms and a set of clauses. We write \mathcal{T}_X for the set of terms and 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
PF	first-order terms over $\mathcal{F}(\Sigma_I)$ and \mathcal{V}_{PG} that do not contain variables whose type is of the form $\tau \rightarrow v$	closures over \mathcal{T}_{PF}
IPG	$\{t \in \mathcal{T}(\Sigma_I, \mathcal{V}_{PG}) \mid t \text{ contains neither type variables nor functional variables}\}$	closures over \mathcal{T}_{IPG}
PG	$\{t \in \mathcal{T}(\Sigma_H, \mathcal{V}_{PG}) \mid t \text{ contains neither type variables nor functional variables}\}$	closures over \mathcal{T}_{PG}
G	$\mathcal{T}(\Sigma_H, \mathcal{V}_H)$	closures over \mathcal{T}_G
H	$\mathcal{T}(\Sigma_H, \mathcal{V}_H)$	constrained clauses over \mathcal{T}_H

5.2.1. First-Order Encodings. The transformation \mathcal{T} from \mathcal{C}_{PF} to \mathcal{C}_F is simply defined as $\mathcal{T}(C \cdot \theta) = C\theta$. We also define a bijective encoding from \mathcal{T}_{IPG} into \mathcal{T}_{PF} and from \mathcal{C}_{IPG} into \mathcal{C}_{PF} . It is very similar to the encoding $\mathcal{F} : \mathcal{T}_{\text{ground}}(\Sigma_H) \rightarrow \mathcal{T}_F$ defined in Definition 3.23, but also encodes variables and does not encode parameters. We reuse the name \mathcal{F} for this new encoding. Potential for confusion is minimal because the two encodings coincide on the values that are in the domain of both.

Definition 5.3 (First-Order Encoding \mathcal{F}). We define $\mathcal{F} : \mathcal{T}_{IPG} \rightarrow \mathcal{T}_{PF}$ recursively as follows: If t is functional, then let t' be the expression obtained by replacing each outermost proper yellow subterm in t by the placeholder symbol \square , and let $\mathcal{F}(t) = \text{fun}_{t'}(\mathcal{F}(\bar{s}_n))$, where \bar{s}_n are the replaced subterms in order of occurrence. If t is a variable x , we define $\mathcal{F}(t) = x$. Otherwise, t is of the form $f(\bar{\tau}) \bar{t}_m$ and we define $\mathcal{F}(t) = f^{\bar{\tau}}(\mathcal{F}(\bar{t}_1), \dots, \mathcal{F}(\bar{t}_m))$.

Applied to a closure $C \cdot \theta \in \mathcal{C}_{IPG}$, the function \mathcal{F} is defined by $\mathcal{F}(C \cdot \theta) = \mathcal{F}(C) \cdot \mathcal{F}(\theta)$, where \mathcal{F} maps each side of each literal and each term in a substitution individually.

Lemma 5.4. *The map \mathcal{F} is a bijection between \mathcal{T}_{IPG} and \mathcal{T}_{PF} and between \mathcal{C}_{IPG} and \mathcal{C}_{PF} .*

Proof. Injectivity of \mathcal{F} can be shown by structural induction. For surjectivity, let $t \in \mathcal{T}_{PF}$. We must show that there exists some $s \in \mathcal{T}_{IPG}$ such that $\mathcal{F}(s) = t$. We proceed by induction on t .

If t is of the form $\text{fun}_{t'}(\bar{t}_n)$, we use the induction hypothesis to derive the existence of some $\bar{s}_n \in \mathcal{T}_{IPG}$ such that $\mathcal{F}(\bar{s}_n) = \bar{t}_n$. Let s be the term resulting from replacing the placeholder symbols \square in t' by \bar{s}_n in order of occurrence. Then $\mathcal{F}(s) = t$.

If t is a variable x , by definition of \mathcal{T}_{PF} , t 's type is not of the form $\tau \rightarrow v$. So, we can set $s = x \in \mathcal{T}_{IPG}$. Then $\mathcal{F}(s) = t$.

If $t = f^{\bar{\tau}}(\bar{t}_n)$, where $f^{\bar{\tau}}$ is not a `fun` symbol, by the induction hypothesis there exist \bar{s}_n such that $\mathcal{F}(\bar{s}_n) = \bar{t}_n$ and set $s = f^{\bar{\tau}}(\bar{s}_n)$. Then $\mathcal{F}(s) = t$.

It follows that \mathcal{F} is also a bijection between \mathcal{C}_{IPG} and \mathcal{C}_{PF} . \square

Lemma 5.5. *For all terms $t \in \mathcal{T}_{IPG}$, all clauses over \mathcal{T}_{IPG} , and all grounding substitutions θ , we have $\mathcal{F}(t)\mathcal{F}(\theta) = \mathcal{F}(t\theta)$ and $\mathcal{F}(C)\mathcal{F}(\theta) = \mathcal{F}(C\theta)$.*

Proof. Since \mathcal{F} maps each side of each literal individually, it suffices to show that $\mathcal{F}(t)\mathcal{F}(\theta) = \mathcal{F}(t\theta)$ for all $t \in \mathcal{T}_{IPG}$. We proceed by structural induction on t .

If t is a variable, the claim is trivial.

If t is nonfunctional and headed by a symbol, the claim follows from the induction hypothesis.

Finally, we consider the case where t is functional. Let t' be the expression obtained by replacing each outermost proper yellow subterm in t by the placeholder symbol \square , and let \bar{s}_n be the replaced subterms in order of occurrence. Since all variables in t are nonfunctional, they must be located in a proper yellow subterm of t , and thus replacing the outermost proper yellow subterm in $t\theta$ by the placeholder symbol \square will result in t' as well. So, using the induction hypothesis, $\mathcal{F}(t)\mathcal{F}(\theta) = \text{fun}_{t'}(\mathcal{F}(\bar{s}_n)\mathcal{F}(\theta)) = \text{fun}_{t'}(\mathcal{F}(\bar{s}_n\theta)) = \mathcal{F}(t\theta)$. \square

Lemma 5.6. *A term $s \in \mathcal{T}_{\text{IPG}}$ is a yellow subterm of $t \in \mathcal{T}_{\text{IPG}}$ if and only if $\mathcal{F}(s)$ is a subterm of $\mathcal{F}(t)$.*

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

Lemma 5.7. *A term $s \in \mathcal{T}_{\text{IPG}}$ is a green subterm of $t \in \mathcal{T}_{\text{IPG}}$ if and only if $\mathcal{F}(s)$ is a green subterm (as defined in Section 3.7) of $\mathcal{F}(t)$.*

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

5.2.2. Indexing of Parameters.

Definition 5.8 (Indexing of Parameters). The transformation \mathcal{J} translates from \mathcal{T}_{PG} to \mathcal{T}_{IPG} by encoding any occurrence of a constant with parameters $f\langle \bar{v} \rangle(\bar{u})$ as $f_{\bar{u}\theta}^{\bar{v}}$, where θ denotes the substitution of the corresponding closure. Formally:

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

We extend \mathcal{J}_θ to clauses by mapping each side of each literal individually. If t is a ground term, the given substitution is irrelevant, so we omit the subscript and simply write $\mathcal{J}(t)$. We extend \mathcal{J} to grounding substitutions by defining $\mathcal{J}(\theta)$ as $x \mapsto \mathcal{J}(x\theta)$. The transformation \mathcal{J} w.r.t. a closure $C \cdot \theta$ is defined as $\mathcal{J}(C \cdot \theta) = \mathcal{J}_\theta(C) \cdot \mathcal{J}(\theta)$.

Lemma 5.9. *For all $t \in \mathcal{T}_{\text{PG}}$, all clauses C over \mathcal{T}_{PG} , and all grounding substitutions θ , we have $\mathcal{J}(t\theta) = \mathcal{J}_\theta(t)\mathcal{J}(\theta)$ and $\mathcal{J}(C\theta) = \mathcal{J}_\theta(C)\mathcal{J}(\theta)$.*

Proof. Since \mathcal{J} maps each side of each literal individually, it suffices to show that $\mathcal{J}(t\theta) = \mathcal{J}_\theta(t)\mathcal{J}(\theta)$. We prove this by induction on the structure of t . If t is a variable, the claim is trivial. For all other cases, the claim follows from the definition of \mathcal{J}_θ and the induction hypothesis. \square

Lemma 5.10. *If $\mathcal{J}_\theta(t)\mathcal{J}(\theta) = \mathcal{J}_\theta(t')\mathcal{J}(\theta)$, then $t\theta = t'\theta$.*

Proof. This becomes clear when viewing \mathcal{J} as composed of two operations: First, we apply the substitution to all variables in parameters. Second, we move type arguments and parameters into indices. Both parts clearly fulfill this lemma's statement. \square

Lemma 5.11. *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{I} (Definition 5.8) and \mathcal{F} (Definitions 3.23 and 5.3), relying on the identification of fun_t and $\text{fun}_{\mathcal{I}(t)}$ (Convention 5.2). For C , the claim holds because \mathcal{I} and \mathcal{F} map each side of each literal individually. \square

5.2.3. Partial Substitution. Let θ be a grounding substitution from \mathcal{V}_H to $\mathcal{T}_{\text{ground}}(\Sigma_H)$. We define a substitution $\mathbf{p}(\theta)$, mapping from \mathcal{V}_H to $\mathcal{T}(\Sigma_H, \mathcal{V}_{PG})$, and a substitution $\mathbf{q}(\theta)$, mapping from \mathcal{V}_{PG} to $\mathcal{T}_{\text{ground}}(\Sigma_H)$, as follows. For each type variable α , let $\alpha\mathbf{p}(\theta) = \alpha\theta$. For each variable $y \in \mathcal{V}_H$, let $y\mathbf{p}(\theta)$ be the term resulting from replacing each nonfunctional yellow subterm at a yellow position p in $y\theta$ by $y_p \in \mathcal{V}_{PG}$. We call these variables y_p the variables introduced by $\mathbf{p}(\theta)$. If a nonfunctional yellow subterm is contained inside another nonfunctional yellow subterm, the outermost nonfunctional yellow subterm is replaced by a variable. The substitution $\mathbf{q}(\theta)$ is defined as $y_p\mathbf{q}(\theta) = y\theta|_p$ for all variables y_p introduced by $\mathbf{p}(\theta)$, and for all other variables $y \in \mathcal{V}_{PG}$, we set $y\mathbf{q}(\theta)$ to be some arbitrary ground term that is independent of θ .

Finally, we define

$$\mathcal{P} : \mathcal{C}_G \rightarrow \mathcal{C}_{PG}, \quad C \cdot \theta \mapsto C\mathbf{p}(\theta) \cdot \mathbf{q}(\theta)$$

For example, if $y\theta = \lambda f(0(ga))$, then $y\mathbf{p}(\theta) = \lambda f(0y_{1.1.1})$ and $y_{1.1.1}\mathbf{q}(\theta) = ga$.

Technically, this definition of \mathbf{p} , \mathbf{q} , and \mathcal{P} depends on the choice of a $\beta\eta$ -normalizer $\downarrow_{\beta\eta}$ because it relies on yellow positions. However, this choice affects the resulting terms and clauses only up to renaming of variables, and our proofs work for any choice of $\downarrow_{\beta\eta}$ as long as we use the same fixed $\downarrow_{\beta\eta}$ for \mathbf{p} , \mathbf{q} , and \mathcal{P} .

Lemma 5.12. *Let θ be a grounding substitution. Then $\mathbf{p}(\theta)\mathbf{q}(\theta) = \theta$.*

Proof. For type variables α , we have $\alpha\mathbf{p}(\theta)\mathbf{q}(\theta) = \alpha\theta$ by definition of \mathbf{p} . We must show that $y\mathbf{p}(\theta)\mathbf{q}(\theta) = y\theta$ for all variables $y \in \mathcal{V}_H$. By definition of \mathbf{p} , $y\mathbf{p}(\theta)$ is obtained from $y\theta$ by replacing each nonfunctional yellow subterm at a yellow position p with y_p . By definition of \mathbf{q} , we have $y_p\mathbf{q}(\theta) = y\theta|_p$ for all such positions p . Therefore, when we apply $\mathbf{q}(\theta)$ to $y\mathbf{p}(\theta)$, we replace each y_p with the original subterm $y\theta|_p$, effectively reconstructing $y\theta$. \square

Lemma 5.13. *Let θ be a substitution from \mathcal{V}_H to $\mathcal{T}_{\text{ground}}(\Sigma_H)$ and ρ be a substitution from \mathcal{V}_{PG} to $\mathcal{T}_{\text{ground}}(\Sigma_H)$. Then $\mathbf{p}(\mathbf{p}(\theta)\rho) = \mathbf{p}(\theta)$.*

Proof. Let $y \in \mathcal{V}_H$. By definition of \mathbf{p} , $y\mathbf{p}(\theta)$ is obtained from $y\theta$ by replacing each nonfunctional yellow subterm at a yellow position p with y_p . Now, consider $y\mathbf{p}(\theta)\rho$. Since ρ is grounding, it will replace each y_p with a ground term. To obtain $y\mathbf{p}(\mathbf{p}(\theta)\rho)$, we take $y\mathbf{p}(\theta)\rho$ and again replace each nonfunctional yellow subterm at a yellow position p with y_p . These positions and the resulting structure will be identical to those in $y\mathbf{p}(\theta)$ because the ground terms introduced by ρ do not affect the overall structure of yellow positions. Therefore, $y\mathbf{p}(\mathbf{p}(\theta)\rho) = y\mathbf{p}(\theta)$ for each variable $y \in \mathcal{V}_H$.

For type variables α , we have $\alpha\mathbf{p}(\theta) = \alpha\theta = \alpha\mathbf{p}(\mathbf{p}(\theta)\rho)$. Thus, we can conclude that $\mathbf{p}(\mathbf{p}(\theta)\rho) = \mathbf{p}(\theta)$. \square

Lemma 5.14. *Let θ be a substitution from \mathcal{V}_H to $\mathcal{T}_{\text{ground}}(\Sigma_H)$. Let ρ be a substitution from \mathcal{V}_{PG} to $\mathcal{T}_{\text{ground}}(\Sigma_H)$ such that $y\rho = y\mathbf{q}(\theta)$ for all y not introduced by $\mathbf{p}(\theta)$. Then $\mathbf{q}(\mathbf{p}(\theta)\rho) = \rho$.*

Proof. Let y_p be a variable introduced by $\mathbf{p}(\mathbf{p}(\theta)\rho)$. By definition of \mathbf{q} , we have $y_p\mathbf{q}(\mathbf{p}(\theta)\rho) = y\mathbf{p}(\theta)\rho|_p$. Moreover, since $y\mathbf{p}(\theta)|_p = y_p$, we have $y\mathbf{p}(\theta)\rho|_p = y_p\rho$. So $y_p\mathbf{q}(\mathbf{p}(\theta)\rho) = y_p\rho$.

Let y be a variable not introduced by $\mathbf{p}(\mathbf{p}(\theta)\rho)$. It remains to show that $y\mathbf{q}(\mathbf{p}(\theta)\rho) = y\rho$. By Lemma 5.13, the variables introduced by $\mathbf{p}(\theta)$ are the same as the variables introduced by $\mathbf{p}(\mathbf{p}(\theta)\rho)$. So y is not introduced by $\mathbf{p}(\theta)$ either. So, by definition of \mathbf{q} , we have $y\mathbf{q}(\mathbf{p}(\theta)\rho) = y\mathbf{q}(\theta)$, and with the assumption of this lemma that $y\rho = y\mathbf{q}(\theta)$, we conclude that $y\mathbf{q}(\mathbf{p}(\theta)\rho) = y\rho$. \square

Lemma 5.15. *Let θ be a grounding substitution. For each variable $y \in \mathcal{V}_H$, $y\mathbf{p}(\theta)$ is the most general term t (unique up to renaming of variables) with the following properties:*

1. *there exists a substitution ρ such that $t\rho = y\theta$;*
2. *t contains no type variables and no functional variables.*

Proof. Let $y \in \mathcal{V}_H$. By Lemma 5.12, $y\mathbf{p}(\theta)$ satisfies property 1, and by definition of \mathbf{p} , it satisfies property 2.

To show that $y\mathbf{p}(\theta)$ is the most general such term, let s be any term satisfying properties 1 and 2, and let σ be a substitution such that $s\sigma = y\theta$. We must show there exists a substitution π such that $y\mathbf{p}(\theta)\pi = s$.

Since s contains no type variables and no functional variables, it is easy to see from the definition of orange subterms that for any orange position p of $s\sigma$, either p is also an orange position of s or there exists a proper prefix q of p such that q is an orange position of s and $s|_q$ is a nonfunctional variable. If there exists such a prefix q , then q must be a nonfunctional yellow position of $s\sigma$ since σ cannot introduce free De Bruijn indices at that position, and thus p cannot be an outermost nonfunctional yellow subterm of $s\sigma$. From these observations, we conclude that any outermost nonfunctional yellow position p of $s\sigma$ must be an orange position of s . In fact, since substituting nonfunctional variables cannot eliminate De Bruijn indices, any outermost nonfunctional yellow position of $s\sigma$ must be a yellow position of s .

Let π map each variable y_p introduced by $\mathbf{p}(\theta)$ to the corresponding term in s at position p . This term exists because p is, by definition of $\mathbf{p}(\theta)$, an outermost nonfunctional yellow position of $y\theta = s\sigma$ and thus, by the above, a yellow position of s . Then $y\mathbf{p}(\theta)\pi = s$ by construction, showing that $y\mathbf{p}(\theta)$ is indeed most general. \square

Lemma 5.16. *Let σ be a substitution, and let ζ be a grounding substitution. Then there exists a substitution π such that $\sigma\mathbf{p}(\zeta) = \mathbf{p}(\sigma\zeta)\pi$ and $\mathbf{q}(\sigma\zeta) = \pi\mathbf{q}(\zeta)$.*

Proof. Let $x \in \mathcal{V}_H$. By Lemma 5.12, $x\sigma\mathbf{p}(\zeta)\mathbf{q}(\zeta) = x\sigma\zeta$. Moreover, $x\sigma\mathbf{p}(\zeta)$ contains only nonfunctional variables. By Lemma 5.15, since $x\mathbf{p}(\sigma\zeta)$ is the most general term with these properties, there must exist a substitution π such that $x\sigma\mathbf{p}(\zeta) = x\mathbf{p}(\sigma\zeta)\pi$. Since the variables in $x_1\mathbf{p}(\sigma\zeta)$ and $x_2\mathbf{p}(\sigma\zeta)$ are disjoint for $x_1 \neq x_2$, we can construct a single substitution π that satisfies $\sigma\mathbf{p}(\zeta) = \mathbf{p}(\sigma\zeta)\pi$, proving the first part of the lemma.

For this construction of π , only the values of π for variables introduced by $\mathbf{p}(\sigma\zeta)$ are relevant. Thus, we can define $y\pi = y\mathbf{q}(\sigma\zeta)$ for all other variables y . Then $y\mathbf{q}(\sigma\zeta) = y\pi\mathbf{q}(\zeta)$ for all variables y not introduced by $\mathbf{p}(\sigma\zeta)$.

Finally, let y_p be a variable introduced by $\mathbf{p}(\sigma\zeta)$. By Lemma 5.12 and the above, $\sigma\zeta = \sigma\mathbf{p}(\zeta)\mathbf{q}(\zeta) = \mathbf{p}(\sigma\zeta)\pi\mathbf{q}(\zeta)$. By Lemma 5.14, $y_p\mathbf{q}(\sigma\zeta) = y_p\pi\mathbf{q}(\zeta)$ for all variables y_p introduced by $\mathbf{p}(\sigma\zeta)$, completing the proof of the second part of the lemma. \square

The redundancy notions of the H level use the map \mathcal{F} , defined in Section 3.7. It is closely related to the maps defined above.

Lemma 5.17. *For all clauses C and grounding substitutions θ ,*

$$\mathcal{F}(C\theta) = \mathcal{T}(\mathcal{F}(\mathcal{I}(\mathcal{P}(C \cdot \theta))))$$

Proof. All of the maps \mathcal{T} , \mathcal{F} , \mathcal{I} , and \mathcal{P} map each literal and each side of a literal individually. So we can focus on one side s of some literal in C , and we must show that

$$\mathcal{F}(s\theta) = \mathcal{F}(\mathcal{I}_{q(\theta)}(sp(\theta)))\mathcal{F}(\mathcal{I}(q(\theta))) \quad (*)$$

By Lemma 5.11,

$$\mathcal{F}(t) = \mathcal{F}(\mathcal{I}(t))$$

for all ground terms $t \in \mathcal{T}_{\text{ground}}(\Sigma_H)$. By Lemma 5.9,

$$\mathcal{I}_\rho(t)\mathcal{I}(\rho) = \mathcal{I}(t\rho)$$

for all $t \in \mathcal{T}_{\text{PG}}$ and grounding substitutions ρ . Also, by Lemma 5.5,

$$\mathcal{F}(t)\mathcal{F}(\rho) = \mathcal{F}(t\rho)$$

for all $t \in \mathcal{T}_{\text{IPG}}$ and grounding substitutions ρ . From the last three equations, we obtain that

$$\mathcal{F}(t\rho) = \mathcal{F}(\mathcal{I}_\rho(t))\mathcal{F}(\mathcal{I}(\rho))$$

for all $t \in \mathcal{T}_{\text{PG}}$ and grounding substitutions ρ .

Using the term s and the substitution θ introduced at the beginning of this proof, take t to be $sp(\theta)$ and ρ to be $q(\theta)$. We then have

$$\mathcal{F}(sp(\theta)q(\theta)) = \mathcal{F}(\mathcal{I}_{q(\theta)}(sp(\theta)))\mathcal{F}(\mathcal{I}(q(\theta)))$$

By Lemma 5.12, this implies (*). □

5.2.4. Grounding. The terms of level H are $\mathcal{T}(\Sigma_H)$. Its clauses \mathcal{C}_H are constrained clauses over these terms. We define the function $\mathcal{G} : \mathcal{C}_H \rightarrow \mathcal{C}_G$ by

$$\mathcal{G}(C[S]) = \{C \cdot \theta \mid \theta \text{ is grounding and } S\theta \text{ is true}\}$$

for each constrained clause $C[S] \in \mathcal{C}_H$.

5.3. Calculi. In this section, we define the calculi PFI_{nf} , IPGI_{nf} , and PGI_{nf} , for the respective levels PF, IPG, and PG. Each of these calculi is parameterized by a relation \succ on ground terms, ground clauses, and closures and by a selection function sel . Based on these parameters, we define the notion of eligibility. The specific requirements on \succ depend on the calculus and are given in the corresponding subsection below. For each of the levels, we define selection functions and the notion of eligibility as follows:

Definition 5.18 (Selection Function). For each level $X \in \{\text{PF}, \text{IPG}, \text{PG}\}$, we define a selection function sel to be a function mapping each closure $C \cdot \theta \in \mathcal{C}_X$ to a subset of C 's literals. We call those literals *selected*. Only negative literals and literals of the form $t \approx \perp$ may be selected.

Definition 5.19 (Eligibility in Closures). Let $X \in \{\text{PF}, \text{IPG}, \text{PG}\}$. Let $C \cdot \theta \in \mathcal{C}_X$. Given a relation \succ and a selection function, a literal $L \in C$ is (*strictly*) *eligible* in $C \cdot \theta$ if it is selected in $C \cdot \theta$ or there are no selected literals in $C \cdot \theta$ and $L\theta$ is (*strictly*) maximal in $C\theta$. A position $L.s.p$ of a closure $C \cdot \theta$ is *eligible* if the literal L is of the form $s \approx t$ with $s\theta \succ t\theta$ and L is either negative and eligible or positive and strictly eligible.

For inferences, we follow the same conventions as in Definition 3.22, but our inference rules operate on closures instead of constrained clauses.

5.3.1. First-Order Levels. The calculus $PFinf^{\succ, sel}$ is parameterized by a relation \succ and a selection function sel . We require that \succ is an admissible term order for $PFinf$ in the following sense:

Definition 5.20. Let \succ be a relation on ground terms, ground clauses, and closures. Such a relation \succ is an *admissible term order for PFinf* if it fulfills the following properties:

- (O1)_{PF} the relation \succ on ground terms is a well-founded total order;
- (O2)_{PF} ground compatibility with contexts: if $s' \succ s$, then $s'[t] \succ s[t]$;
- (O3)_{PF} ground subterm property: $t[s] \succ s$ for ground terms s and t ;
- (O4)_{PF} $u \succ \perp \succ \top$ for all ground terms $u \notin \{\top, \perp\}$;
- (O5)_{PF} $\mathcal{F}(u) \succ \mathcal{F}(u \text{ diff}_{s,t}^{\tau,v})$ for all $s, t, u : \tau \rightarrow v \in \mathcal{T}_{\text{ground}}(\Sigma_I)$;
- (O6)_{PF} the relation \succ on ground clauses is the standard extension of \succ on ground terms via multisets [1, Sect. 2.4];
- (O7)_{PF} for closures $C \cdot \theta$ and $D \cdot \rho$, we have $C \cdot \theta \succ D \cdot \rho$ if and only if $C\theta \succ D\rho$.

We use the notion of green subterms in first-order terms introduced in Section 3.7. We define $x(\rho \cup \theta)$ as $x\rho$ if x occurs in the left premise and as $x\theta$ otherwise.

$$\begin{array}{c}
 \frac{\overbrace{(D' \vee t \approx t') \cdot \rho}^D \quad C \langle u \rangle \cdot \theta}{(D' \vee C \langle t' \rangle) \cdot (\rho \cup \theta)} \text{PFSUP} \quad \frac{\overbrace{(C' \vee u \not\approx u') \cdot \theta}^C}{C' \cdot \theta} \text{PFEQRES} \\
 \\
 \frac{\overbrace{(C' \vee u' \approx v' \vee u \approx v) \cdot \theta}^C}{(C' \vee v \not\approx v' \vee u \approx v') \cdot \theta} \text{PFEQFACT} \quad \frac{(C' \vee s \approx t) \cdot \theta}{(C' \vee D) \cdot \theta} \text{PFCLAUSIFY} \\
 \\
 \frac{C \langle u \rangle \cdot \theta}{(C \langle \perp \rangle \vee u \approx \top) \cdot \theta} \text{PFBOOLHOIST} \quad \frac{C \langle u \rangle \cdot \theta}{(C \langle \top \rangle \vee u \approx \perp) \cdot \theta} \text{PFLOOBHOIST} \\
 \\
 \frac{\overbrace{(C' \vee s \approx t) \cdot \theta}^C}{C' \cdot \theta} \text{PFFALSEELIM} \\
 \\
 \frac{\overbrace{(C' \vee \mathcal{F}(s) \approx \mathcal{F}(s')) \cdot \theta}^C}{C' \vee \mathcal{F}(s \text{ diff}_{u,v}^{\tau,v}) \approx \mathcal{F}(s' \text{ diff}_{u,v}^{\tau,v}) \cdot \theta} \text{PFDARGCONG} \\
 \\
 \frac{C \langle \mathcal{F}(u) \rangle \cdot \theta}{C \langle \mathcal{F}(v) \rangle \vee \mathcal{F}(u \text{ diff}_{u\theta,v\theta}^{\tau,v}) \not\approx \mathcal{F}(v \text{ diff}_{u\theta,v\theta}^{\tau,v}) \cdot \rho} \text{PFEEXT} \\
 \\
 \frac{}{\mathcal{F}(u \text{ diff}_{u\theta,v\theta}^{\tau,v}) \not\approx \mathcal{F}(v \text{ diff}_{u\theta,v\theta}^{\tau,v}) \vee \mathcal{F}(u s) \approx \mathcal{F}(v s) \cdot \theta} \text{PFDIFF}
 \end{array}$$

Side conditions for PFSUP:

1. $t\rho = u\theta$;
2. u is not a variable;
3. u is nonfunctional;

4. $t\rho \succ t'\rho$;
5. $D\rho \prec C[u]\theta$;
6. the position of u is eligible in $C \cdot \theta$;
7. $t \approx t'$ is strictly eligible in $D \cdot \rho$;
8. if $t'\rho$ is Boolean, then $t'\rho = \top$.

Side conditions for PFEQRES:

1. $u\theta = u'\theta$;
2. $u \not\approx u'$ is eligible in $C \cdot \theta$.

Side conditions for PFEQFACT:

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

Side conditions for PFCLAUSIFY:

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

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

Side conditions for PFBOOLHOIST and PFLOOBHOIST:

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

Side conditions for PFFALSEELIM:

1. $(s \approx t)\theta = \perp \approx \top$;
2. $s \approx t$ is strictly eligible in $C \cdot \theta$.

Side conditions for PFARGCONG:

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 $C \cdot \theta$.

Side conditions for PFEEXT:

1. the position of $\mathcal{F}(u)$ is eligible in $C \cdot \theta$;
2. the type of u is $\tau \rightarrow v$;
3. $v \in \mathcal{T}_{\text{IPG}}$ is a term of type $\tau \rightarrow v$ whose nonfunctional yellow subterms are different variables and the variables in $\mathcal{F}(v)$ do not occur in $C[\mathcal{F}(u)]$.
4. $u\theta \succ v\rho$;
5. ρ is a grounding substitution that coincides with θ on all variables in $C[\mathcal{F}(u)]$.

Side conditions for PFDIFF:

1. τ and v are ground types;
2. $u, v, s \in \mathcal{T}_{\text{IPG}}$ are terms whose nonfunctional yellow subterms are different fresh variables;
3. θ is a grounding substitution.

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

Definition 5.21. Let \succ be a relation on $\mathcal{T}_{\text{ground}}(\Sigma_I)$, on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_I)$, and on closures \mathcal{C}_{IPG} . Such a relation \succ is an *admissible term order for IPGInf* if it fulfills the following properties:

- (O1)_{IPG} the relation \succ on ground terms is a well-founded total order;
- (O2)_{IPG} 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)_{IPG} ground yellow subterm property: $t \ll s \gg \succeq s$ for ground terms s and t ;
- (O4)_{IPG} $u \succ \perp \succ \top$ for all ground terms $u \notin \{\top, \perp\}$;
- (O5)_{IPG} $u \succ u \text{ diff}_{s,t}^{\tau,v}$ for all ground terms $s, t, u : \tau \rightarrow v$.
- (O6)_{IPG} the relation \succ on ground clauses is the standard extension of \succ on ground terms via multisets [1, Sect. 2.4];
- (O7)_{IPG} for closures $C \cdot \theta$ and $D \cdot \rho$, we have $C \cdot \theta \succ D \cdot \rho$ if and only if $C\theta \succ D\rho$.

The rules of $\text{IPGInf}^{\succ, \text{sel}}$ (abbreviated IPGInf) are the following. We assume that for the binary inference IPGSUP , the premises do not have any variables in common, and we define $x(\rho \cup \theta)$ as $x\rho$ if x occurs in the left premise and as $x\theta$ otherwise.

$$\frac{\overbrace{D' \vee t \approx t' \cdot \rho}^D \quad C \langle u \rangle \cdot \theta}{D' \vee C \langle t' \rangle \cdot (\rho \cup \theta)} \text{IPGSUP}$$

with the following side conditions:

1. $t\rho = u\theta$;
2. u is not a variable;
3. u is nonfunctional;
4. $t\rho \succ t'\rho$;
5. $D\rho \prec C\langle u \rangle \theta$;
6. the position of u is eligible in $C \cdot \theta$;
7. $t \approx t'$ is strictly eligible in $D \cdot \rho$;
8. if $t'\rho$ is Boolean, then $t'\rho = \top$.

$$\frac{\overbrace{C' \vee u \not\approx u' \cdot \theta}^C}{C' \cdot \theta} \text{IPGEQRES} \quad \frac{\overbrace{C' \vee u' \approx v' \vee u \approx v \cdot \theta}^C}{C' \vee v \not\approx v' \vee u \approx v' \cdot \theta} \text{IPGEQFACT}$$

Side conditions for IPGEQRES:

1. $u\theta = u'\theta$;
2. $u \not\approx u'$ is eligible in $C \cdot \theta$.

Side conditions for IPGEQFACT:

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

$$\frac{C' \vee s \approx t \cdot \theta}{C' \vee D \cdot \theta} \text{IPGCLAUSIFY}$$

with the following side conditions:

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

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

$$\frac{C \langle u \rangle \cdot \theta}{C \langle \perp \rangle \vee u \approx \top \cdot \theta} \text{IPGBOOLHOIST} \quad \frac{C \langle u \rangle \cdot \theta}{C \langle \top \rangle \vee u \approx \perp \cdot \theta} \text{IPGLOOBHOIST}$$

each with the following side conditions:

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

$$\frac{\overbrace{C' \vee s \approx t \cdot \theta}^C}{C' \cdot \theta} \text{IPGFALSEELIM}$$

with the following side conditions:

1. $(s \approx t)\theta = \perp \approx \top$;
2. $s \approx t$ is strictly eligible in $C \cdot \theta$.

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

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 $C \cdot \theta$.

$$\frac{C\langle u \rangle \cdot \theta}{C\langle v \rangle \vee u \text{diff}_{u\theta, v\rho}^{\tau, v} \not\approx v \text{diff}_{u\theta, v\rho}^{\tau, v} \cdot \rho} \text{IPGEXT}$$

with the following side conditions:

1. the position of u is eligible in $C\langle u \rangle \cdot \theta$;
2. the type of u is $\tau \rightarrow v$;
3. $v \in \mathcal{T}_{\text{IPG}}$ is a term whose nonfunctional yellow subterms are different fresh variables;
4. $u\theta \succ v\rho$;
5. ρ is a grounding substitution that coincides with θ on all variables in $C\langle u \rangle$.

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

with the following side conditions:

1. τ and v are ground types;
2. $u, v, s \in \mathcal{T}_{\text{IPG}}$ are terms whose nonfunctional yellow subterms are different variables;
3. θ is a grounding substitution.

5.3.3. Partly Substituted Ground Higher-Order Level. Like on the other levels, the calculus PGInf is parameterized by a relation \succ and a selection function sel .

Definition 5.22. Let \succ be a relation on $\mathcal{T}_{\text{ground}}(\Sigma_H)$, on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_H)$, and on \mathcal{C}_{PG} . Such a relation \succ is an *admissible term order for PGInf* if it fulfills the following properties:

- (O1)_{PG} the relation \succ on ground terms is a well-founded total order;
- (O2)_{PG} ground compatibility with yellow contexts: $s' \succ s$ implies $t \ll s' \succ t \ll s$ for ground terms s , s' , and t ;
- (O3)_{PG} ground yellow subterm property: $t \ll s \succeq s$ for ground terms s and t ;
- (O4)_{PG} $u \succ \perp \succ \top$ for all ground terms $u \notin \{\top, \perp\}$;
- (O5)_{PG} $u \succ u \text{diff}\langle \tau, v \rangle(s, t)$ for all ground terms $s, t, u : \tau \rightarrow v$.
- (O6)_{PG} the relation \succ on ground clauses is the standard extension of \succ on ground terms via multisets [1, Sect. 2.4];
- (O7)_{PG} for closures $C \cdot \theta$ and $D \cdot \rho$, we have $C \cdot \theta \succ D \cdot \rho$ if and only if $C\theta \succ D\rho$.

The calculus rules of PGInf are a verbatim copy of those of IPGInf , with the following exceptions:

- PGInf uses Σ_H instead of Σ_I and \mathcal{C}_{PG} instead of \mathcal{C}_{IPG} .
- The rules are prefixed by PG instead of IPG.
- PGARGCONG uses $\text{diff}\langle \tau, v \rangle(u, v)$ instead of $\text{diff}_{u,v}^{\tau,v}$.
- PGEXT uses $\text{diff}\langle \tau, v \rangle(u, v)$ instead of $\text{diff}_{u\theta, v\rho}^{\tau, v}$.
- PGDIFF uses $\text{diff}\langle \tau, v \rangle(u, v)$ instead of $\text{diff}_{u\theta, v\theta}^{\tau, v}$.

5.4. Redundancy Criteria and Saturation. In this subsection, we define redundancy criteria for the levels PF, IPG, PG, and H 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 5.6 to lift refutational completeness from level PF to level H.

Definition 5.23. A set N of clauses is called *saturated up to redundancy* if every inference with premises in N is redundant w.r.t. N .

5.4.1. First-Order Level. In this subsection, let \succ be an admissible term order for $PFInf$ (Definition 5.20), and let $pfsel$ be a selection function on C_{PF} (Definition 5.18).

We define a notion of variable-irreducibility, roughly following Nieuwenhuis and Rubio [24] and Bachmair et al. [3] (where it is called “order-irreducibility”):

Definition 5.24. A closure literal $L \cdot \theta \in C_{PF}$ is *variable-irreducible* w.r.t. a ground term rewrite system R if, for all variables x in L , $x\theta$ is irreducible w.r.t. the rules $s \rightarrow t \in R$ with $L\theta \succ s \approx t$ and all Boolean subterms of $x\theta$ are either \top or \perp . A closure $C \cdot \theta \in C_{PF}$ is *variable-irreducible* w.r.t. R if all its literals are variable-irreducible w.r.t. R . Given a set N of closures, we write $\text{irred}_R(N)$ for the set of variable-irreducible closures in N w.r.t. R .

Remark 5.25. The restriction that $L\theta \succ s \approx t$ cannot be replaced by $C\theta \succ s \approx t$ because we need it to ensure that $\text{irred}_R(N)$ is saturated when N is.

Here is an example demonstrating that variable-irreducibility would not be closed under inferences if we used the entire clause for comparison: Let $e \succ d \succ c \succ b \succ a$ and $R = \{e \rightarrow b\}$. Then a notion replacing the restriction $L\theta \succ s \approx t$ by $C\theta \succ s \approx t$ would say that $x \approx a \vee x \approx b \llbracket x \equiv e \rrbracket$ and $e \approx c \vee e \approx d$ are order-irreducible w.r.t. R . By SUP (second literal of the first clause into the second literal of the second clause), we obtain

$$x \approx a \vee e \approx c \vee b \approx d \llbracket x \equiv e \rrbracket$$

Now $x\theta$ in the first literal has become order-reducible by $e \rightarrow b$ since $e \approx b$ is now smaller than the largest literal $e \approx c$ of the clause.

Definition 5.26 (Inference Redundancy). Given $\iota \in PFInf$ and $N \subseteq C_{PF}$, let $\iota \in PFRed_1(N)$ if for all confluent term rewrite systems R oriented by \succ whose only Boolean normal forms are \top and \perp such that $\text{concl}(\iota)$ is variable-irreducible, we have $R \cup O \models_{o\lambda} \text{concl}(\iota)$, where $O = \text{irred}_R(N)$ if ι is a DIFF inference, and $O = \{E \in \text{irred}_R(N) \mid E \prec \text{mprem}(\iota)\}$ otherwise.

To connect to the redundancy criteria of the higher levels, we need to establish a connection to the $FInf$ inference system defined in Section 3.7:

Lemma 5.27. Let $\iota_{PF} \in PFInf^{\succ, pfsel}$. Let $C_1 \cdot \theta_1, \dots, C_m \cdot \theta_m$ be its premises and $C_{m+1} \cdot \theta_{m+1}$ its conclusion. Then

$$\frac{C_1 \theta_1 \cdots C_m \theta_m}{C_{m+1} \theta_{m+1}}$$

is a valid $FInf^\succ$ inference ι_F , and the rule names of ι_{PF} and ι_F correspond up to the prefixes PF and F.

Proof. This is easy to see by comparing the rules of $PFInf$ and $FInf$. It is crucial that the concepts of eligibility match: If a literal or a position is (strictly) eligible in a closure $C \cdot \theta \in C_{PF}$ (according to the PF concept of eligibility), then the corresponding literal or position is (strictly) eligible in $C\theta$ (according to the F concept of eligibility). \square

5.4.2. Indexed Partly Substituted Ground Higher-Order Level. In this subsubsection, let \succ be an admissible term order for $IPGInf$ (Definition 5.21), and let $ipgsel$ be a selection function on C_{IPG} (Definition 5.18).

To lift the notion of inference redundancy, we need to connect the inference systems $PFInf$ and $IPGInf$ as follows. Since the mapping \mathcal{F} is bijective (Lemma 5.4), we can transfer the order \succ from the IPG level to the PF level:

Definition 5.28. Based on \succ , we define a relation $\succ_{\mathcal{F}}$ on ground terms T_F , ground clauses C_F , and closures C_{PF} as $d \succ_{\mathcal{F}} e$ if and only if $\mathcal{F}^{-1}(d) \succ \mathcal{F}^{-1}(e)$ for all terms, clauses, or closures d and e .

Lemma 5.29. Since \succ is an admissible term order for $IPGInf$ (Definition 5.21), the relation $\succ_{\mathcal{F}}$ is an admissible term order for $PFInf$ (Definition 5.20).

Proof. This is easy to see, considering that \mathcal{F} is a bijection between $T_{\text{ground}}(\Sigma_I)$ and T_{PF}^{gnd} (Lemma 5.4), that higher-order yellow subterms and first-order subterms correspond by Lemma 5.6, that \mathcal{F} maps each side of each literal individually, and that $\mathcal{F}(C)\mathcal{F}(\theta) = \mathcal{F}(C\theta)$ for all $C \cdot \theta \in C_{IPG}$ (Lemma 5.5). \square

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

Definition 5.30. Based on $ipgsel$, we define $\mathcal{F}(ipgsel)$ as a selection function that selects the literals of $C \in C_{PF}$ corresponding to the $ipgsel$ -selected literals in $\mathcal{F}^{-1}(C)$.

Definition 5.31. We extend \mathcal{F} to inference rules by mapping an inference $\iota \in IPGInf$ to the inference

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

Lemma 5.32. The mapping \mathcal{F} is a bijection between $IPGInf^{\succ, ipgsel}$ and $PFInf^{\succ_{\mathcal{F}}, \mathcal{F}(ipgsel)}$.

Proof. This is easy to see by comparing the rules of $IPGInf$ and $PFInf$ and considering Remark ???. It is crucial that the following concepts match:

- Green subterms on the PF level correspond to green subterms on the IPG level by Lemma 5.7.
- The term orders correspond (Definition 5.28).
- The selected literals correspond; i.e., a literal L is selected in a closure $C \cdot \theta$ if and only if the literal $\mathcal{F}(L)$ is selected in $\mathcal{F}(C \cdot \theta)$. This follows directly from the definition of $\mathcal{F}(ipgsel)$ (Definition 5.30).
- The concepts of eligibility correspond; i.e., a literal L of a closure $C \cdot \theta \in C_{IPG}$ is (strictly) eligible w.r.t. \succeq if and only if the literal $\mathcal{F}(L)$ of the closure $\mathcal{F}(C \cdot \theta)$ is (strictly) eligible w.r.t. $\succeq_{\mathcal{F}}$; and a position $L.s.p$ of a closure $C \cdot \theta \in C_{IPG}$ is eligible w.r.t. \succeq if and only if the position $\mathcal{F}(L).\mathcal{F}(s).q$ of the closure $\mathcal{F}(C \cdot \theta)$ is eligible w.r.t. $\succeq_{\mathcal{F}}$, where q is the position corresponding to p . This is true because eligibility (Definition 5.19) depends only on the selected literals and the term order, which correspond as discussed above. \square

Definition 5.33. Given a term rewrite system R on T_F , we say that a closure $C \cdot \theta \in C_{IPG}$ is variable-irreducible w.r.t. R and \succ if $\mathcal{F}(C \cdot \theta)$ is variable-irreducible w.r.t. R and $\succ_{\mathcal{F}}$. We write $\text{irred}_R(N)$ for all variable-irreducible closures in a set $N \subseteq C_{IPG}$.

Definition 5.34 (Inference Redundancy). Given $\iota \in IPGInf^{\succ, ipgsel}$ and $N \subseteq C_{IPG}$, let $\iota \in IPGRed_1(N)$ if for all confluent term rewrite systems R on T_F oriented by $\succ_{\mathcal{F}}$ whose

only Boolean normal forms are \top and \perp such that $\text{concl}(\iota)$ is variable-irreducible w.r.t. R , we have

$$R \cup O \models_{\text{o}\lambda} \mathcal{F}(\text{concl}(\iota))$$

where $O = \text{irred}_R(\mathcal{F}(N))$ if ι is a IPGDIFF inference, and $O = \{E \in \text{irred}_R(\mathcal{F}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(\text{mprem}(\iota))\}$ otherwise.

Lemma 5.35. *Let $\iota_{\text{IPG}} \in \text{IPGInf}^{\succ, \text{ipgsel}}$. Let $C_1 \cdot \theta_1, \dots, C_m \cdot \theta_m$ be its premises and $C_{m+1} \cdot \theta_{m+1}$ its conclusion. Then*

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

is a valid $F\text{Inf}^{\succ_{\mathcal{F}}}$ inference ι_F , and the rule names of ι_{IPG} and ι_F correspond up to the prefixes IPG and F.

Proof. By Lemma 5.32, we know that $\mathcal{F}(\iota_{\text{IPG}})$ is a valid $P\text{FInf}^{\succ_{\mathcal{F}}, \mathcal{F}(\text{ipgsel})}$ inference, and the rule names coincide up to the prefixes PF and IPG.

Now, applying Lemma 5.27 to $\mathcal{F}(\iota_{\text{IPG}})$ and using the fact that $\mathcal{F}(C_i)\mathcal{F}(\theta_i) = \mathcal{F}(C_i\theta_i)$ (Lemma 5.5), we obtain that

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

is a valid $F\text{Inf}^{\succ_{\mathcal{F}}}$ inference ι_F , and the rule names of $\mathcal{F}(\iota_{\text{IPG}})$ and ι_F correspond up to the prefixes PF and F.

Combining these two results, we conclude that the rule names of ι_{IPG} and ι_F correspond up to the prefixes IPG and F, which completes the proof. \square

Using the bijection between IPGInf and PFInf , we can show that saturation w.r.t. IPGInf implies saturation w.r.t. PFInf :

Lemma 5.36. *Let N be saturated up to redundancy w.r.t. $\text{IPGInf}^{\succ, \text{ipgsel}}$. Then $\mathcal{F}(N)$ is saturated up to redundancy w.r.t. $\text{PFInf}^{\succ_{\mathcal{F}}, \mathcal{F}(\text{ipgsel})}$.*

Proof. By Lemma 5.32 because the notions of inference redundancy correspond. \square

5.4.3. Partly Substituted Ground Higher-Order Level. In this subsubsection, let \succ be an admissible term order for PGInf (Definition 5.22), and let pgsel be a selection function on \mathcal{C}_{PG} (Definition 5.18).

Since mapping \mathcal{I} is clearly bijective for ground terms and ground clauses, we can transfer \succ from the PG level to the IPG level as follows:

Definition 5.37. Let \succ be a relation on $\mathcal{T}_{\text{ground}}(\Sigma_H)$, on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_H)$, and on closures \mathcal{C}_{PG} . We define a relation $\succ_{\mathcal{I}}$ on $\mathcal{T}_{\text{ground}}(\Sigma_I)$ and on clauses over $\mathcal{T}_{\text{ground}}(\Sigma_I)$ as $d \succ_{\mathcal{I}} e$ if and only if $\mathcal{I}^{-1}(d) \succ \mathcal{I}^{-1}(e)$ for all terms or clauses d and e . For closures $C \cdot \theta, D \cdot \rho \in \mathcal{C}_{\text{IPG}}$, we define $C \cdot \theta \succ_{\mathcal{I}} D \cdot \rho$ if $C\theta \succ_{\mathcal{I}} D\rho$.

Lemma 5.38. *Since \succ is an admissible term order for PGInf (Definition 5.22), the relation $\succ_{\mathcal{I}}$ is an admissible term order for IPGInf (Definition 5.21).*

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

Lemma 5.39. *Given $D \cdot \rho, C \cdot \theta \in \mathcal{C}_{\text{PG}}$, we have $D \cdot \rho \succ C \cdot \theta$ if and only if $\mathcal{I}(D \cdot \rho) \succ_{\mathcal{I}} \mathcal{I}(C \cdot \theta)$.*

Proof. By $(O7)_{\text{IPG}}$, $D \cdot \rho \succ C \cdot \theta$ if and only if $C\theta \succ D\rho$. By Definition 5.37, this is equivalent to $\mathcal{I}(D\rho) \succ_{\mathcal{I}} \mathcal{I}(C\theta)$. Since $\mathcal{I}(D\rho) = \mathcal{I}_\rho(D)\mathcal{I}(\rho)$ and $\mathcal{I}(C\theta) = \mathcal{I}_\theta(C)\mathcal{I}(\theta)$ by Lemma 5.9, this is equivalent to $\mathcal{I}_\rho(D)\mathcal{I}(\rho) \succ_{\mathcal{I}} \mathcal{I}_\theta(C)\mathcal{I}(\theta)$. By $(O7)_{\text{IPG}}$, this is equivalent to $\mathcal{I}(D \cdot \rho) \succ_{\mathcal{I}} \mathcal{I}(C \cdot \theta)$. \square

Definition 5.40. Given a term rewrite system R on \mathcal{T}_F , we say that a closure $C \cdot \theta \in \mathcal{C}_{\text{PG}}$ is variable-irreducible w.r.t. R if $\mathcal{F}(\mathcal{I}(C \cdot \theta))$ is. We write $\text{irred}_R(N)$ for all variable-irreducible closures in a set $N \subseteq \mathcal{C}_{\text{PG}}$.

Definition 5.41 (Inference Redundancy). Let $N \subseteq \mathcal{C}_{\text{PG}}$. Let $\iota \in \text{PGInf}$ an inference with premises $C_1 \cdot \theta_1, \dots, C_m \cdot \theta_m$ and conclusion $C_{m+1} \cdot \theta_{m+1}$. We define $\iota \in \text{PGRed}_I(N)$ if

1. the inference ι' given as

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

is not a valid $FInf^{\succ_{\mathcal{I}F}}$ inference such that the names of ι and ι' correspond up to the prefixes PG and F; or

2. for all confluent term rewrite systems R oriented by $\succ_{\mathcal{I}F}$ whose only Boolean normal forms are \top and \perp such that $C_{m+1} \cdot \theta_{m+1}$ is variable-irreducible, we have

$$R \cup O \models_{\text{o}\lambda} \mathcal{F}(\mathcal{I}(C_{m+1} \cdot \theta_{m+1}))$$

where $O = \text{irred}_R(\mathcal{F}(\mathcal{I}(N)))$ if ι is a PGDIFF inference and $O = \{E \in \text{irred}_R(\mathcal{F}(\mathcal{I}(N))) \mid E \prec_{\mathcal{I}F} \mathcal{F}(\mathcal{I}(C_m\theta_m))\}$ if ι is some other inference.

We transfer the selection function $pgsel$ as follows:

Definition 5.42. Let $N \subseteq \mathcal{C}_{\text{PG}}$ be a set of closures. Then we choose a function \mathcal{I}_N^{-1} , depending on this set N , such that $\mathcal{I}_N^{-1}(C) \in N$ and $\mathcal{I}(\mathcal{I}_N^{-1}(C)) = C$ for all $C \in \mathcal{I}(N)$. Then we define $\mathcal{I}(pgsel, N)$ as a selection function that selects the literals of $C \in \mathcal{I}(N)$ corresponding to the $pgsel$ -selected literals in $\mathcal{I}_N^{-1}(C)$ and that selects arbitrary literals in all other closures.

Lemma 5.43. *Let $N \subseteq \mathcal{C}_{\text{PG}}$ be saturated up to redundancy w.r.t. $\text{PGInf}^{\succ, pgsel}$. Then $\mathcal{I}(N)$ is saturated up to redundancy w.r.t. $\text{IPGInf}^{\succ_{\mathcal{I}}, \mathcal{I}(pgsel, N)}$.*

Proof. Let ι' be a $\text{IPGInf}^{\succ_{\mathcal{I}}, \mathcal{I}(pgsel, N)}$ inference from $\mathcal{I}(N)$. We must show that $\iota' \in \text{IPGRed}_I(\mathcal{I}(N))$. It suffices to construct a PGInf inference ι with premises $\mathcal{I}_N^{-1}(\text{prems}(\iota'))$ such that $\mathcal{I}(\text{concl}(\iota)) = \text{concl}(\iota')$ and the rule names of ι and ι' coincide up to the prefixes PG and IPG. Then, by saturation, $\iota \in \text{PGRed}_I(N)$; i.e., condition 1 or condition 2 of Definition 5.41, is satisfied. Condition 1 cannot be satisfied because it contradicts Lemma 5.35 applied to ι' . Thus, condition 2 must be satisfied. Then, by Definition 5.34, $\iota' \in \text{IPGRed}_I(\mathcal{I}(N))$.

Finding such an inference ι is straightforward for all inference rules. We illustrate it with the rule IPGEQRES: Let ι' be

$$\frac{C' \vee u \not\approx u' \cdot \theta}{C' \cdot \theta} \text{IPGEQRES}$$

Then there exists a corresponding $PGInf$ inference ι from $\mathcal{I}_N^{-1}(C' \vee u \not\approx u' \cdot \theta)$. (See Definition 5.42 for the definition of \mathcal{I}_N^{-1} .) The eligibility condition is fulfilled because the term order, and the selections are transferred according to Definitions 5.37 and 5.42 and Lemma 5.39. The equality condition is fulfilled by Lemma 5.10. The conclusion $concl(\iota)$ of this inference has the property $\mathcal{I}(concl(\iota)) = concl(\iota')$, as desired. \square

5.4.4. Full Higher-Order Level. In this subsubsection, let \succ be an admissible term order (Definition 3.16) and let $hsel$ be a selection function (Definition 3.18). We extend \succ to closures C_{PG} by $C \cdot \theta \succ D \cdot \rho$ if and only if $C\theta \succ D\rho$. Then we can use it for $PGInf$ as well:

Lemma 5.44. *The relation \succ is an admissible term order for $PGInf$.*

Proof. Conditions (O1) to (O6) are identical to conditions $(O1)_{PG}$ to $(O6)_{PG}$. Condition $(O7)_{PG}$ is fulfilled by the given extension of \succ to closures. \square

Definition 5.45. Given a term rewrite system R on \mathcal{T}_F , we say that a closure $C \cdot \theta \in C_G$ is variable-irreducible w.r.t. R if $\mathcal{F}(\mathcal{I}(\mathcal{P}(C \cdot \theta)))$ is. We write $irred_R(N)$ for all variable-irreducible closures in a set $N \subseteq C_G$.

For the H level, we define both clause and inference redundancy. Below, we write $\mathcal{FIIPG}(C)$ for $\mathcal{F}(\mathcal{I}(\mathcal{P}(G(C))))$ and $\mathcal{FIIP}(C)$ for $\mathcal{F}(\mathcal{I}(\mathcal{P}(C)))$.

Definition 5.46 (Clause Redundancy). Given a constrained clause $C \in C_H$ and a set $N \subseteq C_H$, let $C \in HRed_C(N)$ if for all confluent term rewrite systems R on \mathcal{T}_F oriented by $\succ_{\mathcal{IF}}$ whose only Boolean normal forms are \top and \perp and all $C' \in irred_R(\mathcal{FIIPG}(C))$, at least one of the following two conditions holds:

1. $R \cup \{E \in irred_R(\mathcal{FIIPG}(N)) \mid E \prec_{\mathcal{IF}} C'\} \models_{o\lambda} C'$; or
2. there exists clauses $D \in N$ and $D' \in irred_R(\mathcal{FIIPG}(D))$ such that $C \sqsupseteq D$ and $\mathcal{T}(D') = \mathcal{T}(C')$.

Definition 5.47 (Inference Redundancy). Let $N \subseteq C_H$. Let $\iota \in HInf$ an inference with premises $C_1[S_1], \dots, C_m[S_m]$ and conclusion $C_{m+1}[S_{m+1}]$. We define $HRed_I$ so that $\iota \in HRed_I(N)$ if for all substitutions $(\theta_1, \dots, \theta_{m+1})$ for which ι is rooted in $FInf$ (Definition 3.31), and for all confluent term rewrite systems R oriented by $\succ_{\mathcal{IF}}$ whose only Boolean normal forms are \top and \perp such that $C_{m+1} \cdot \theta_{m+1}$ is variable-irreducible, we have

$$R \cup O \models_{o\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$$

where $O = irred_R(\mathcal{FIIPG}(N))$ if ι is a DIFF inference and $O = \{E \in irred_R(\mathcal{FIIPG}(N)) \mid E \prec_{\mathcal{IF}} \mathcal{F}(C_m\theta_m)\}$ if ι is some other inference.

The selection function is transferred in a similar way as with \mathcal{I} :

Definition 5.48. Let $N \subseteq C_G$. We choose a function \mathcal{P}_N^{-1} , depending on this set N , such that $\mathcal{P}_N^{-1}(C) \in N$ and $\mathcal{P}(\mathcal{P}_N^{-1}(C)) = C$ for all $C \in \mathcal{P}(N)$. Similarly, for $N \subseteq C_H$, 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 $\mathcal{PG}(hsel, N)$ as a selection function for C_{PG} as follows: Given a clause $C_{PG} \in \mathcal{PG}(N)$, let $C_H \cdot \theta = \mathcal{P}_{G(N)}^{-1}(C_{PG})$ and $C_H[S] = \mathcal{G}_N^{-1}(C_H \cdot \theta)$. We define $\mathcal{PG}(hsel, N)$ to select $L_{PG} \in C_{PG}$ if and only if there exists a literal L_H selected in $C_H[S]$ by $hsel$ such that $L_{PG} = L_H p(\theta)$. Given a clause $C_{PG} \notin \mathcal{PG}(N)$, $\mathcal{PG}(hsel, N)$ can select arbitrary literals.

Lemma 5.49. *Let R be a confluent term rewrite system on \mathcal{T}_{PF} oriented by $\succ_{\mathcal{F}}$ whose only Boolean normal forms are \top and \perp . Let $C \cdot \theta, D \cdot \rho \in \mathcal{CPG}$. Let $C \cdot \theta$ be variable-irreducible w.r.t. R . Let σ be a substitution such that $z\theta = z\sigma\rho$ for all variables z in C and $D = C\sigma$. Then $D \cdot \rho$ is variable-irreducible w.r.t. R .*

Proof. Let L be a literal in $\mathcal{F}(\mathcal{J}_\rho(D))$. We must show that for all variables x in L , $\mathcal{F}(\mathcal{J}(x\rho))$ is irreducible w.r.t. the rules $s \rightarrow t \in R$ with $L\mathcal{F}(\mathcal{J}(\rho)) \succ_{\mathcal{F}} s \approx t$ and all Boolean subterms of $\mathcal{F}(\mathcal{J}(x\rho))$ are either \top or \perp . Then there exists a literal $L_0 \in D$ such that $L = \mathcal{F}(\mathcal{J}_\rho(L_0))$. Let x be a variable in L . Then it is also a variable in L_0 , occurring outside of parameters. Since $D = C\sigma$, x occurs in $C\sigma$ outside of parameters. Since $C \cdot \theta \in \mathcal{CPG}$, the clause C contains only nonfunctional variables, and thus a literal $L'_0 \in C$ with $L'_0\sigma = L_0$ must contain a variable z outside of parameters such that x occurs outside of parameters in $z\sigma$. So $\mathcal{F}(\mathcal{J}(x\rho))$ is a subterm of $\mathcal{F}(\mathcal{J}(z\sigma\rho)) = \mathcal{F}(\mathcal{J}(z\theta))$. Let $L' = \mathcal{F}(\mathcal{J}_\theta(L'_0)) \in \mathcal{F}(\mathcal{J}_\theta(C))$. Then L' also contains z . By variable-irreducibility of $C \cdot \theta$, $\mathcal{F}(\mathcal{J}(z\theta))$ is irreducible w.r.t. the rules $s \rightarrow t \in R$ with $L'\mathcal{F}(\mathcal{J}(\theta)) \succ_{\mathcal{F}} s \approx t$ and all Boolean subterms of $\mathcal{F}(\mathcal{J}(z\theta))$ are either \top or \perp . Then the subterm $\mathcal{F}(\mathcal{J}(x\rho))$ of $\mathcal{F}(\mathcal{J}(z\theta))$ is also irreducible w.r.t. the rules $s \rightarrow t \in R$ with $L'\mathcal{F}(\mathcal{J}(\theta)) \succ_{\mathcal{F}} s \approx t$ and all Boolean subterms of $\mathcal{F}(\mathcal{J}(x\rho))$ are either \top or \perp . It remains to show that $L'\mathcal{F}(\mathcal{J}(\theta)) = L\mathcal{F}(\mathcal{J}(\rho))$. We have $L'\mathcal{F}(\mathcal{J}(\theta)) = \mathcal{F}(\mathcal{J}_\theta(L'_0))\mathcal{F}(\mathcal{J}(\theta)) = \mathcal{F}(\mathcal{J}_\theta(L'_0\theta)) = \mathcal{F}(\mathcal{J}_\theta(L'_0\sigma\rho)) = \mathcal{F}(\mathcal{J}_\rho(L_0\rho)) = \mathcal{F}(\mathcal{J}_\rho(L_0))\mathcal{F}(\mathcal{J}(\rho)) = L\mathcal{F}(\mathcal{J}(\rho))$, using Lemma 5.5 and Lemma 5.9. \square

Lemma 5.50 (Lifting of Order Conditions). *Let $t[\![T]\!]$ and $s[\![S]\!]$ be constrained terms over $\mathcal{T}(\Sigma_H)$, and let ζ be a grounding substitution such that $S\zeta$ and $T\zeta$ are true. If $t\zeta \succ s\zeta$, then $t[\![T]\!] \not\leq s[\![S]\!]$. The same holds for constrained literals.*

Proof. We prove the contrapositive. If $t[\![T]\!] \leq s[\![S]\!]$, then, by (O7), $t\zeta \preceq s\zeta$. Therefore, since \succ is asymmetric by (O1), $t\zeta \not\succ s\zeta$. The proof for constrained literals is analogous, using (O6) and (O8). \square

Lemma 5.51 (Lifting of Maximality Conditions). *Let $C[\![S]\!] \in \mathcal{CH}$. 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 $C[\![S]\!]$ such that $L\theta = L_0$.*

Proof. By Definition 3.17, a literal L of a constrained clause $C[\![S]\!]$ is maximal if for all $K \in C$ such that $K[\![S]\!] \succeq L[\![S]\!]$, we have $K[\![S]\!] \preceq L[\![S]\!]$.

Since $L_0 \in C\theta$, there exist literals L in $C[\![S]\!]$ 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 constrained literals by (O9) and transitivity implies existence of maximal elements in nonempty finite sets. Let K be a literal in $C[\![S]\!]$ such that $K[\![S]\!] \succeq L[\![S]\!]$. We must show that $K[\![S]\!] \preceq L[\![S]\!]$. By Lemma 5.50, $K\theta \not\prec L\theta = L_0$. 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 $K[\![S]\!] \preceq L[\![S]\!]$ because we chose L to be maximal among all literals in $C[\![S]\!]$ 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 5.52 (Lifting of Eligibility). *Let $N \subseteq \mathcal{CH}$. Let $C_{\text{PG}} \cdot \theta_{\text{PG}} \in \mathcal{PG}(N)$, let $C_H \cdot \theta = \mathcal{P}_{\mathcal{G}(N)}^{-1}(C_{\text{PG}} \cdot \theta_{\text{PG}})$ and let $C_H[\![S]\!] = \mathcal{G}_N^{-1}(C_H \cdot \theta)$.*

- Let L_{PG} be a literal in $C_{\text{PG}} \cdot \theta_{\text{PG}}$ that is (strictly) eligible w.r.t. $\mathcal{PG}(\text{hsel}, N)$. Then there exists a literal L_H in C_H such that $L_{\text{PG}} = L_H \mathbf{p}(\theta)$ and, given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in $C_H[\![S]\!]$, L_H is (strictly) eligible in $C_H[\![S]\!]$ w.r.t. σ and hsel .

- Let $L_{\text{PG}} \cdot s_{\text{PG}} \cdot p_{\text{PG}}$ be a green position of $C_{\text{PG}} \cdot \theta_{\text{PG}}$ that is eligible w.r.t. $\mathcal{PG}(h\text{sel}, N)$. Then there exists a green position $L_{\text{H}} \cdot s_{\text{H}} \cdot p_{\text{H}}$ of C_{H} such that
 - $L_{\text{PG}} = L_{\text{H}} \mathbf{p}(\theta)$;
 - $s_{\text{PG}} = s_{\text{H}} \mathbf{p}(\theta)$;
 - $* \ p_{\text{PG}} = p_{\text{H}}$, or
 - * $p_{\text{PG}} = p_{\text{H}} \cdot q$ for some nonempty q , the subterm u_{H} at position $L_{\text{H}} \cdot s_{\text{H}} \cdot p_{\text{H}}$ of C_{H} is not a variable but is variable-headed, and $u_{\text{H}} \theta$ is nonfunctional; and
 - given substitutions σ and ζ with $x\theta = x\sigma\zeta$ for all variables x in $C_{\text{H}}[\![S]\!]$, $L_{\text{H}} \cdot s_{\text{H}} \cdot p_{\text{H}}$ is eligible in $C_{\text{H}}[\![S]\!]$ w.r.t. σ and $h\text{sel}$.

Proof. Let L_{PG} be a literal in $C_{\text{PG}} \cdot \theta_{\text{PG}}$ that is (strictly) eligible w.r.t. $\mathcal{PG}(h\text{sel}, N)$. By the definition of eligibility (Definition 5.19), there are two ways to be (strictly) eligible:

- L_{PG} is selected by $\mathcal{PG}(h\text{sel}, N)$. By Definition 5.48, there exists a literal L_{H} selected by $h\text{sel}$ such that $L_{\text{PG}} = L_{\text{H}} \mathbf{p}(\theta)$. By Definition 3.19, L_{H} is (strictly) eligible in $C_{\text{H}}[\![S]\!]$ w.r.t. σ because it is selected.
- There are no selected literals in $C_{\text{PG}} \cdot \theta_{\text{PG}}$ and $L_{\text{PG}} \theta_{\text{PG}}$ is (strictly) maximal in $C_{\text{PG}} \theta_{\text{PG}}$. By Definition 5.48, there are no selected literals in $C_{\text{H}}[\![S]\!]$. Since $C_{\text{PG}} \theta_{\text{PG}} = C_{\text{H}} \theta = C_{\text{H}} \sigma \zeta$, by Lemma 5.51, there exists a literal $L_{\text{H}} \in C_{\text{H}}$ such that $L_{\text{H}} \sigma$ is (strictly) maximal in $C_{\text{H}} \sigma$. By Definition 3.19, L_{H} is (strictly) eligible in $C_{\text{H}}[\![S]\!]$ w.r.t. σ .

For the second part of the lemma, let $L_{\text{PG}} \cdot s_{\text{PG}} \cdot p_{\text{PG}}$ be a green position of $C_{\text{PG}} \cdot \theta_{\text{PG}}$ that is eligible w.r.t. $\mathcal{PG}(h\text{sel}, N)$. By Definition 5.19, the literal L_{PG} is of the form $s_{\text{PG}} \approx t_{\text{PG}}$ with $s_{\text{PG}} \theta_{\text{PG}} \succ t_{\text{PG}} \theta_{\text{PG}}$ and L_{PG} 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 $C_{\text{H}}[\![S]\!]$ w.r.t. σ and $h\text{sel}$ such that $L_{\text{PG}} = L_{\text{H}} \mathbf{p}(\theta)$. Then L_{H} must be of the form $s_{\text{H}} \approx t_{\text{H}}$ with $s_{\text{PG}} = s_{\text{H}} \mathbf{p}(\theta)$ and $t_{\text{PG}} = t_{\text{H}} \mathbf{p}(\theta)$. Since $s_{\text{PG}} \theta_{\text{PG}} \succ t_{\text{PG}} \theta_{\text{PG}}$, we have $s_{\text{H}} \not\leq t_{\text{H}}$. By Definition 3.19, every green position in $L_{\text{H}} \cdot s_{\text{H}}$ is eligible in $C_{\text{H}}[\![S]\!]$ w.r.t. σ and $h\text{sel}$.

It remains to show that there exists a green position $L_{\text{H}} \cdot s_{\text{H}} \cdot p_{\text{H}}$ in C_{H} such that either $p_{\text{PG}} = p_{\text{H}}$ or $p_{\text{PG}} = p_{\text{H}} \cdot q$ for some nonempty q , the subterm u_{H} at position $L_{\text{H}} \cdot s_{\text{H}} \cdot p_{\text{H}}$ of C_{H} is not a variable but variable-headed, and $u_{\text{H}} \theta$ is nonfunctional.

Since p_{PG} is a green position of $s_{\text{PG}} = s_{\text{H}} \mathbf{p}(\theta)$, position p_{PG} 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 $p_{\text{H}} = p_{\text{PG}}$. In the second case, let p_{H} be the position of the variable-headed term. Then $p_{\text{H}} \cdot q = p_{\text{PG}}$ for some nonempty q . Moreover, since p_{PG} is a green position of s_{PG} , the subterm of s_{PG} at position p_{H} , which is $u_{\text{H}} \theta$, cannot be functional. Since $\mathbf{p}(\theta)$ maps nonfunctional variables to nonfunctional variables, the subterm of s_{H} at position p_{H} cannot be a variable, but it is variable-headed. \square

Lemma 5.53 (Lifting Lemma). *Let $N \subseteq C_{\text{H}}$ be saturated up to redundancy w.r.t. $H\text{Inf}^{\succ, h\text{sel}}$. Then $\mathcal{PG}(N)$ is saturated up to redundancy w.r.t. $PG\text{Inf}^{\succ, \mathcal{PG}(h\text{sel}, N)}$.*

Proof. Let ι_{PG} be a $PG\text{Inf}$ inference from $\mathcal{PG}(N)$. We must show that $\iota_{\text{PG}} \in PG\text{Red}_1(\mathcal{PG}(N))$. It suffices to construct a $H\text{Inf}$ inference ι_{H} from N such that ι_{H} and ι_{PG} are of the form

$$\frac{C_1[\![S_1]\!] \quad \cdots \quad C_m[\![S_m]\!]}{C_{m+1}[\![S_{m+1}]\!]} \iota_{\text{H}} \quad \frac{\mathcal{P}(C_1 \cdot \theta_1) \quad \cdots \quad \mathcal{P}(C_m \cdot \theta_m)}{E \cdot \xi} \iota_{\text{PG}} \quad (*)$$

for some $C_1[\![S_1]\!], \dots, C_{m+1}[\![S_{m+1}]\!] \in \mathcal{C}_H$, $E \cdot \xi \in \mathcal{C}_{PG}$, and grounding substitutions $\theta_1, \dots, \theta_{m+1}$ such that $S_1\theta_1, \dots, S_{m+1}\theta_{m+1}$ are true and $C_{m+1}\mathbf{p}(\theta_{m+1}) = E\pi$ and $x\pi\mathbf{q}(\theta_{m+1}) = x\xi$ for some substitution π and all variables x in E .

Here is why this suffices: By Definition 5.41, it suffices to show that for all confluent term rewrite systems R on \mathcal{T}_F oriented by $\succ_{\mathcal{T}_F}$ whose only Boolean normal forms are \mathbf{T} and \mathbf{L} , such that $E \cdot \xi$ is variable-irreducible w.r.t. R , we have

$$R \cup O \models_{o\lambda} \mathcal{F}(\mathcal{J}(E \cdot \xi))$$

where $O = \text{irred}_R(\mathcal{F}(\mathcal{J}(\mathcal{G}(N))))$ if ι_{PG} is a DIFF inference, and otherwise $O = \{E \in \text{irred}_R(\mathcal{F}(\mathcal{J}(\mathcal{G}(N)))) \mid E \prec \mathcal{F}(\mathcal{J}(\mathcal{P}(C_m \cdot \theta_m)))\}$. Let R be such a rewrite system. By Lemma 5.49, since $E \cdot \xi$ is variable-irreducible, using $E\pi = C_{m+1}\mathbf{p}(\theta_{m+1})$, and $x\xi = x\pi\mathbf{q}(\theta_{m+1})$, also $C_{m+1}\mathbf{p}(\theta_{m+1}) \cdot \mathbf{q}(\theta_{m+1})$ and thus $C_{m+1} \cdot \theta_{m+1}$ are variable-irreducible w.r.t. R . By saturation, $\iota_H \in HRed_1(N)$, and thus, by definition of $HRed_1$, (Definition 5.47), it suffices to show that ι_H is rooted in $FInf$ for $(\theta_1, \dots, \theta_{m+1})$ (Definition 3.31), which holds by Lemma 5.11, Definition 5.41 and the fact that $C_{m+1}\theta_{m+1} = C_{m+1}\mathbf{p}(\theta_{m+1})\mathbf{q}(\theta_{m+1}) = E\pi\mathbf{q}(\theta_{m+1}) = E\xi$.

For most rules, the following special case of $(*)$ suffices: We construct a $HInf$ inference ι_H from N such that ι_H and ι_{PG} are of the form

$$\frac{C_1[\![S_1]\!] \quad \dots \quad C_m[\![S_m]\!] \quad \iota_H}{C'_{m+1}\sigma [\![S'_{m+1}]\!]} \quad \frac{\mathcal{P}(C_1 \cdot \theta_1) \quad \dots \quad \mathcal{P}(C_m \cdot \theta_m)}{C'_{m+1}\mathbf{p}(\sigma\zeta) \cdot \xi} \quad \iota_{PG} \quad (**)$$

for some $C_1[\![S_1]\!], \dots, C_m[\![S_m]\!], C'_{m+1}[\![S'_{m+1}]\!] \in \mathcal{C}_H$ and substitutions $\theta_1, \dots, \theta_m, \zeta, \sigma, \xi$ such that $S_1\theta_1, \dots, S_m\theta_m, S'_{m+1}\zeta$ are true and $x\xi = x\mathbf{q}(\sigma\zeta)$ for all variables x in $C'_{m+1}\mathbf{p}(\sigma\zeta)$.

Here is why this is a special case of $(*)$ with $C_{m+1} = C'_{m+1}\sigma$, $S_{m+1} = S'_{m+1}$, $E = C'_{m+1}\mathbf{p}(\sigma\zeta)$: By Lemma 5.16, there exists a substitution π such that $\sigma\mathbf{p}(\zeta) = \mathbf{p}(\sigma\zeta)\pi$ and $\mathbf{q}(\sigma\zeta) = \pi\mathbf{q}(\zeta)$. Thus, $C_{m+1}\mathbf{p}(\zeta) = C'_{m+1}\sigma\mathbf{p}(\zeta) = C'_{m+1}\mathbf{p}(\sigma\zeta)\pi = E\pi$ and $x\pi\mathbf{q}(\zeta) = x\mathbf{q}(\sigma\zeta) = x\xi$ for all variables x in E , as required for $(*)$.

PGSUP: If ι_{PG} is a PG SUP inference

$$\frac{\overbrace{D'_{PG} \vee t_{PG} \approx t'_{PG} \cdot \rho_{PG}}^{D_{PG}} \quad C_{PG} \langle u_{PG} \rangle \cdot \theta_{PG}}{D'_{PG} \vee C_{PG} \langle t'_{PG} \rangle \cdot (\rho_{PG} \cup \theta_{PG})} \text{PGSUP}$$

then we construct a corresponding SUP or FLUIDSUP inference ι_H . Let $D_H \cdot \rho_H = \mathcal{P}_{\mathcal{G}(N)}^{-1}(D_{PG} \cdot \rho_{PG})$ and $C_H \cdot \theta_H = \mathcal{P}_{\mathcal{G}(N)}^{-1}(C_{PG} \cdot \theta_{PG})$. (See Definition 5.48 for the definition of $\mathcal{P}_{\mathcal{G}(N)}^{-1}$.) Let $D_H[\![T]\!] = \mathcal{G}_N^{-1}(D_H \cdot \rho_H)$ and $C_H[\![S]\!] = \mathcal{G}_N^{-1}(C_H \cdot \theta_H)$. (See Definition 5.48 for the definition of \mathcal{G}_N^{-1} .) These clauses $D_H[\![T]\!]$ and $C_H[\![S]\!]$ will be the premises of ι_H . Condition 7 of PG SUP states that $t_H \approx t'_H$ is strictly eligible in $D_{PG} \cdot \rho_{PG}$. Let $D_H = D'_H \vee t_H \approx t'_H$, where $t_H \approx t'_H$ is the literal that Lemma 5.51 guarantees to be strictly eligible with $t_H\mathbf{p}(\theta_H) = t_{PG}$ and $t'_H\mathbf{p}(\theta_H) = t'_{PG}$. Condition 8 of PG SUP states that if $t'_{PG}\rho_{PG}$ is Boolean, then $t'_{PG}\rho_{PG} = \mathbf{T}$. Thus, there are no selected literals in $D_{PG} \cdot \rho_{PG}$. By Definition 5.48, it follows that there are no selected literals in $D_H[\![T]\!] \cdot \rho_H$ (condition 7 of SUP or FLUIDSUP) and that $t_H \approx t'_H$ is strictly maximal (condition 6 of SUP or FLUIDSUP). Let $L_{PG}.s_{PG}.p_{PG}$ be the position of the green subterm u_{PG} in C_{PG} . Condition 6 of PG SUP states that $L_{PG}.s_{PG}.p_{PG}$ is eligible in $C_{PG} \cdot \theta_{PG}$. Let $L_H.s_H.p_H$ be the position in C_H that Lemma 5.52 guarantees to be eligible

in $C_H[S]$ w.r.t. any suitable σ (condition 5 of SUP or FLUIDSUP). Let u_H be the subterm of C_H at position $L_H.s_H.p_H$. By Lemma 5.52, one of the following cases applies:

CASE 1: $p_{PG} = p_H$. Then we construct a SUP inference.

Lemma 5.52 tells us that $s_{PG} = s_H p(\theta_H)$ and thus $u_{PG} = u_H p(\theta_H)$ because $p_{PG} = p_H$. By condition 1 of PGSUP, $t_{PG}\rho_{PG} = u_{PG}\theta_{PG}$. It follows that $t_H\rho_H = t_H p(\rho_H)q(\rho_H) = t_{PG}\rho_{PG} = u_{PG}\theta_{PG} = u_H p(\theta_H)q(\theta_H) = u_H\theta_H$. Let $\rho_H \cup \theta_H$ be the substitution that coincides with ρ_H on all variables in $D_H[T]$ and with θ_H on all other variables. Then $\rho_H \cup \theta_H$ is a unifier of t_H and u_H . Moreover, by construction, $T\rho_H$ and $S\theta_H$ are true. Thus, by definition of CSU^{upto} (Definition 3.11), there exists $(\sigma, U) \in CSU^{upto}(T, S, t_H \equiv u_H)$ (condition 1 of SUP) and a substitution ζ such that $U\zeta$ is true and $x\sigma\zeta = x(\rho_H \cup \theta_H)$ for all relevant variables x .

Conditions 2 and 3 of PGSUP state that $u_{PG} = u_H p(\theta_H)$ is not a variable and nonfunctional. Thus, by definition of p , u_H is not a variable and $u_H\sigma$ is nonfunctional (conditions 2 and 3 of SUP).

By Lemma 5.50, the condition $(t_H[T]) \not\leq (t'_H[T])$ (condition 4 of SUP) follows from condition 4 of PGSUP.

Finally, the given inference ι_{PG} and the constructed inference ι_H are of the form $(**)$ with $C_1[S_1] = D_H[T]$, $C_2[S_2] = C_H[S]$, $C'_3 = D'_H \vee C_H \langle t'_H \rangle_{p_H} S'_3 = U$, $\theta_1 = \rho_H$, $\theta_2 = \theta_H$, and $\xi = \rho_{PG} \cup \theta_{PG}$.

CASE 2: $p_{PG} = p_H.q$ for some nonempty q , u_H is not a variable but is variable-headed, and $u_H\theta$ is nonfunctional. Then we construct a FLUIDSUP inference.

Lemma 5.52 tells us that $s_{PG} = s_H p(\theta_H)$. Thus, the subterm of s_{PG} at position p_H is $u_H p(\theta_H)$. So q is a green position of $u_H p(\theta_H)$, and the subterm at that position is u_{PG} —i.e., $u_H p(\theta_H) = (u_H p(\theta_H)) \langle u_{PG} \rangle_q$. Let $v = \lambda(u_H p(\theta_H)) \langle n \rangle_q$, where n is the appropriate De Bruijn index to refer to the initial λ .

Let z be a fresh variable (condition 8 of FLUIDSUP). We define θ'_H by $z\theta'_H = v\theta_{PG}$, $x\theta'_H = x\rho_H$ for all variables x in $D_H[T]$ and $x\theta'_H = x\theta_H$ for all other variables x . Then, using condition 1 of PGSUP, $z t_H \theta'_H = v\theta_{PG} (t_H \rho_H) = v\theta_{PG} (t_{PG}\rho_{PG}) = v\theta_{PG} (u_{PG}\theta_{PG}) = (v u_{PG})\theta_{PG} = (u_H p(\theta_H)) \langle u_{PG} \rangle_q \theta_{PG} = u_H p(\theta_H)\theta_{PG} = u_H\theta_H = u_H\theta'_H$. So θ'_H is a unifier of $z t_H$ and u_H . Thus, by definition of CSU (Definition 3.13), there exists a unifier $\sigma \in CSU(z t_H \equiv u_H)$ and a substitution ζ such that $x\sigma\zeta = x\theta'_H$ for all relevant variables x (condition 1 of FLUIDSUP).

By the assumption of this case, u_H is not a variable but is variable-headed (condition 2 of FLUIDSUP).

Since q is a green position of $u_H p(\theta_H)$, the type of $u_H p(\theta_H)$ and the type of $u_H\sigma$ is nonfunctional (condition 3 of FLUIDSUP).

By Lemma 5.50, the condition $t_H[T] \not\leq t'_H[T]$ (condition 4 of FLUIDSUP) follows from condition 4 of PGSUP.

By condition 4 of PGSUP, $t_H\rho_H = t_{PG}\rho_{PG} \neq t'_H\rho_{PG} = t'_H\rho_H$. Thus, $(z t_H)\sigma\zeta = v\theta_{PG} (t_H \rho_H) = (u_H p(\theta_H))\theta_{PG} \langle t_H \rho_H \rangle_q \neq (u_H p(\theta_H))\theta_{PG} \langle t'_H \rho_H \rangle_q = v\theta_{PG} (t'_H \rho_H) = (z t'_H)\sigma\zeta$. So, $(z t'_H)\sigma \neq (z t_H)\sigma$ (condition 9 of FLUIDSUP).

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

The inferences ι_{PG} and ι_H are of the form $(*)$ with $C_1[S_1] = D_H[T]$, $C_2[S_2] = C_H[S]$, $C_3[S_3] = (D'_H \vee C_H \langle z t'_H \rangle_{p_H} [T, S])\sigma$, $\theta_1 = \rho_H$, $\theta_2 = \theta_H$, $\theta_3 = \zeta$, $E = D'_{PG} \vee C_{PG} \langle t'_{PG} \rangle_{p_{PG}}$, and $\xi = \rho_{PG} \cup \theta_{PG}$. In the following, we elaborate why $C_3p(\theta_3) = E\pi$ and $x\pi q(\theta_3) = x\xi$ for some substitution π and all variables x in E , as required for $(*)$. We invoke Lemma 5.16 to

obtain a substitution π such that $\sigma\mathbf{p}(\zeta) = \mathbf{p}(\sigma\zeta)\pi$ and $\mathbf{q}(\sigma\zeta) = \pi\mathbf{q}(\zeta)$. Then

$$\begin{aligned}
C_3\mathbf{p}(\theta_3) &= (D'_H \vee C_H \langle z t'_H \rangle_{p_H})\sigma\mathbf{p}(\zeta) \\
&= (D'_H\sigma\mathbf{p}(\zeta) \vee C_H\sigma\mathbf{p}(\zeta)\langle(z t'_H)\sigma\mathbf{p}(\zeta)\rangle_{p_H}) \\
&\stackrel{(1)}{=} (D'_H\sigma\mathbf{p}(\zeta) \vee C_H\sigma\mathbf{p}(\zeta)\langle(z t_H)\sigma\mathbf{p}(\zeta)\langle t'_H\sigma\mathbf{p}(\zeta)\rangle_q\rangle_{p_H}) \\
&\stackrel{(2)}{=} (D'_H\sigma\mathbf{p}(\zeta) \vee C_H\sigma\mathbf{p}(\zeta)\langle u_H\sigma\mathbf{p}(\zeta)\langle t'_H\sigma\mathbf{p}(\zeta)\rangle_q\rangle_{p_H}) \\
&\stackrel{(3)}{=} (D'_H\sigma\mathbf{p}(\zeta) \vee C_H\sigma\mathbf{p}(\zeta)\langle t'_H\sigma\mathbf{p}(\zeta)\rangle_{p_{PG}}) \\
&= (D'_H\mathbf{p}(\sigma\zeta)\pi \vee C_H\mathbf{p}(\sigma\zeta)\pi\langle t'_H\mathbf{p}(\sigma\zeta)\pi\rangle_{p_{PG}}) \\
&= (D'_H\mathbf{p}(\sigma\zeta) \vee C_H\mathbf{p}(\sigma\zeta)\langle t'_H\mathbf{p}(\sigma\zeta)\rangle_{p_{PG}})\pi \\
&= (D'_{PG} \vee C_{PG}\langle t'_{PG}\rangle_{p_{PG}})\pi \\
&= E\pi
\end{aligned}$$

Step (1) can be justified as follows: By Lemma 5.12, $z\sigma\mathbf{p}(\zeta)\mathbf{q}(\zeta) = v\theta_{PG}$. Since $z\sigma\mathbf{p}(\zeta)$ and $v\theta_{PG}$ contain only nonfunctional variables, $z\sigma\mathbf{p}(\zeta)$ must be a λ -abstraction whose λ binds exactly one De Bruijn index, which is located at orange position $1.q$ w.r.t. $\downarrow_{\beta\eta long}$. Thus, $(z t_H)\sigma\mathbf{p}(\zeta)$ and $(z t'_H)\sigma\mathbf{p}(\zeta)$ are identical up to the subterm at green position q , which is $t_H\sigma\mathbf{p}(\zeta)$ and $t'_H\sigma\mathbf{p}(\zeta)$ respectively. For step (2), we use that $(z t_H)\sigma = u_H\sigma$. For step (3), we use that u_H is the green subterm of C_H at position p_H .

PGEQRES: If ι_{PG} is an PGEQRES inference

$$\frac{\overbrace{C'_{PG} \vee u_{PG} \not\approx u'_{PG} \cdot \theta_{PG}}^{C_{PG}}}{C'_{PG} \cdot \theta_{PG}} \text{PGEQRES}$$

, then we construct a corresponding EQRES inference ι_H . Let $C_H \cdot \theta_H = \mathcal{P}_{G(N)}^{-1}(C_{PG} \cdot \theta_{PG})$ and $C_H[S] = G_N^{-1}(C_H \cdot \theta_H)$. This clause $C_H[S]$ will be the premise of ι_H .

A condition of PGEQRES is that $u_{PG} \not\approx u'_{PG}$ is eligible in $C_{PG} \cdot \theta_{PG}$. Let $C_H = C'_H \vee u_H \not\approx u'_H$, where $u_H \not\approx u'_H$ is the literal that Lemma 5.52 guarantees to be eligible w.r.t. any suitable σ , with $u_H\mathbf{p}(\theta_H) = u_{PG}$ and $u'_H\mathbf{p}(\theta_H) = u'_{PG}$.

Another condition of PGEQRES states that $u_{PG}\theta_{PG} = u'_{PG}\theta_{PG}$. Thus, $u_H\theta_H = u'_H\theta_H$, and therefore there exists $(\sigma, U) \in \text{CSU}^{\text{upto}}(S, u_H \equiv u'_H)$ and a substitution ζ such that $U\zeta$ is true and $x\sigma\zeta = x\theta_H$ for all variables x in $C_H[S]$.

Finally, the given inference ι_{PG} and the constructed inference ι_H are of the form (**) with $C_1[S_1] = C_H[S]$, $C'_2 = C'_H$, $S'_2 = U$, $\theta_1 = \theta_H$, and $\xi = \theta_{PG}$.

PGEQFACT: If ι_{PG} is an PGEQFACT inference

$$\frac{\overbrace{C'_{PG} \vee u'_{PG} \approx v'_{PG} \vee u_{PG} \approx v_{PG} \cdot \theta_{PG}}^{C_{PG}}}{C'_{PG} \vee v_{PG} \not\approx v'_{PG} \vee u_{PG} \approx v'_{PG} \cdot \theta_{PG}} \text{PGEQFACT}$$

then we construct a corresponding EQFACT inference ι_H . Let $C_H \cdot \theta_H = \mathcal{P}_{G(N)}^{-1}(C_{PG} \cdot \theta_{PG})$ and $C_H[S] = G_N^{-1}(C_H \cdot \theta_H)$. This clause $C_H[S]$ will be the premise of ι_H .

A condition of PGEQFACT is that $u_{PG} \approx v_{PG} \cdot \theta_{PG}$ is maximal in $C_{PG} \cdot \theta_{PG}$. Let $u_H \approx v_H$ be the literal that Lemma 5.51 guarantees to be maximal in $C_H[S]$ w.r.t. any

suitable σ , with $u_H p(\theta_H) = u_{PG}$ and $v_H p(\theta_H) = v_{PG}$. Choose C'_H , u'_H , and v'_H such that $C_H = C'_H \vee u'_H \approx v'_H \vee u_H \approx v_H$, $C'_H p(\theta_H) = C'_{PG}$, $u'_H p(\theta_H) = u'_{PG}$, and $v'_H p(\theta_H) = v'_{PG}$.

Another condition of PGEQFACT states that there are no selected literals in $C_{PG} \cdot \theta_{PG}$. By Definition 5.48, it follows that there are no selected literals in $C_H[S]$.

Another condition of PGEQFACT states that $u_{PG} \theta_{PG} = u'_{PG} \theta_{PG}$. Thus, $u_H \theta_H = u'_H \theta_H$, and therefore there exists $(\sigma, U) \in CSU^{\text{upto}}(S, u_H \equiv u'_H)$ and a substitution ζ such that $U\zeta$ is true and $x\sigma\zeta = x\theta_H$ for all variables x in $C_H[S]$.

The last condition of PGEQFACT is that $u_{PG} \theta_{PG} \succ v_{PG} \theta_{PG}$ —i.e., $u_H \sigma \zeta \succ v_H \sigma \zeta$. By Lemma 5.50, $(u_H[S])\sigma \not\leq (v_H[S])\sigma$.

Finally, the given inference ι_{PG} and the constructed inference ι_H are of the form (**) with $C_1[S_1] = C_H[S]$, $C'_2 = C'_H \vee v_H \not\approx v'_H \vee u_H \approx v'_H$, $S'_2 = U$, $\theta_1 = \theta_H$, and $\xi = \theta_{PG}$.

PGCLAUSIFY: If ι_{PG} is a PGCLAUSIFY inference

$$\frac{\overbrace{C'_{PG} \vee s_{PG} \approx t_{PG} \cdot \theta_{PG}}^{C_{PG}}}{C'_{PG} \vee D_{PG} \cdot \theta_{PG}} \text{PGCLAUSIFY}$$

with τ_{PG} being the type and u_{PG} and v_{PG} being the terms used for condition 2. Then we construct a corresponding CLAUSIFY inference ι_H . Let $C_H \cdot \theta_H = \mathcal{P}_{G(N)}^{-1}(C_{PG} \cdot \theta_{PG})$ and $C_H[S] = \mathcal{G}_N^{-1}(C_H \cdot \theta_H)$. This clause $C_H[S]$ will be the premise of ι_H .

Condition 1 of PGCLAUSIFY is that $s_{PG} \approx t_{PG}$ is strictly eligible in $C_{PG} \cdot \theta_{PG}$. Let $C_H = C'_H \vee s_H \approx t_H$, where $s_H \approx t_H$ is the literal that Lemma 5.52 guarantees to be strictly eligible w.r.t. any suitable σ (condition 2 of CLAUSIFY), with $s_H p(\theta_H) = s_{PG}$ and $t_H p(\theta_H) = t_{PG}$.

Comparing the listed triples in PGCLAUSIFY and CLAUSIFY, we see that there must be a triple (s'_H, t'_H, D_H) listed for CLAUSIFY such that $(s'_H \rho, t'_H \rho, D_H \rho) = (s_{PG}, t_{PG} \theta_{PG}, D_{PG})$ with $\rho = \{\alpha \mapsto \tau_{PG}, x \mapsto u_{PG}, y \mapsto v_{PG}\}$ is the triple used for ι_{PG} (condition 4 of CLAUSIFY). Inspecting the listed triples, we see that s_{PG} cannot be a variable and that $s_{PG} \theta_{PG} = s_H \theta_H$ is of Boolean type. It follows that s_H is not a variable (condition 3 of CLAUSIFY) because if it were, then, by definition of p , $s_H p(\theta_H) = s_{PG}$ would be a variable.

Moreover, we observe that $s_H \theta_H = s_{PG} \theta_{PG} = s'_H \rho \theta_{PG}$ and $t_H \theta_H = t_{PG} \theta_{PG} = t'_H \rho = t'_H \rho \theta_{PG}$. Thus the substitution mapping all variables x in s'_H and t'_H to $x\rho \theta_{PG}$ and all other variables x to $x\theta_H$ is a unifier of $s_H \equiv s'_H$ and $t_H \equiv t'_H$. So there exists a unifier $\sigma \in CSU(s_H \equiv s'_H, t_H \equiv t'_H)$ (condition 1 of CLAUSIFY) and a substitution ζ such that $x\sigma\zeta = x\theta_H$ for all variables x in $C_H[S]$.

Finally, the given inference ι_{PG} and the constructed inference ι_H are of the form (**) with $C_1[S_1] = C_H[S]$, $C'_2 = C'_H \vee D_H$, $S'_2 = S\sigma$, $\theta_1 = \theta_H$, and $\xi = \theta_{PG}$.

PGBOOLHOIST: If ι_{PG} is a PGBOOLHOIST inference

$$\frac{C_{PG} \langle u_{PG} \rangle \cdot \theta_{PG}}{C_{PG} \langle \perp \rangle \vee u_{PG} \approx \top \cdot \theta_{PG}} \text{PGBOOLHOIST}$$

then we construct a corresponding BOOLHOIST or FLUIDBOOLHOIST inference ι_H . Let $C_H \cdot \theta_H = \mathcal{P}_{G(N)}^{-1}(C_{PG} \cdot \theta_{PG})$ and $C_H[S] = \mathcal{G}_N^{-1}(C_H \cdot \theta_H)$. This clause $C_H[S]$ will be the premise of ι_H . Let $L_{PG}.s_{PG}.p_{PG}$ be the position of u_{PG} in C_{PG} . Condition 3 of PGBOOLHOIST states that $L_{PG}.s_{PG}.p_{PG}$ is eligible in $C_{PG} \cdot \theta_{PG}$. Let $L_H.s_H.p_H$ be the position that Lemma 5.52 guarantees to be eligible in $C_H[S]$ w.r.t. any suitable σ (condition 3 of BOOLHOIST or

condition 8 of FLUIDBOOLHOIST). Let u_H be the subterm at position $L_H.s_H.p_H$ in $C_H[S]$. By Lemma 5.52, one of the following cases applies:

CASE 1: $p_{PG} = p_H$. Then we construct a BOOLHOIST inference.

Lemma 5.52 tells us that $s_{PG} = s_H \mathfrak{p}(\theta_H)$ and thus $u_{PG} = u_H \mathfrak{p}(\theta_H)$ because $p_{PG} = p_H$. By Condition 1 of PGBOOLHOIST, $u_{PG} = u_H \mathfrak{p}(\theta_H)$ is of Boolean type and thus $u_H \theta_H$ is of Boolean type. So there exists a most general type substitution σ such that u_H is Boolean (condition 1 of BOOLHOIST) and a substitution ζ such that $x\sigma\zeta = x\theta_H$ for all variables x in $C_H[S]$.

By condition 2 of PGBOOLHOIST, $u_{PG} = u_H \mathfrak{p}(\theta_H)$ is not a variable and is neither \top nor \perp . By condition 1 of PGBOOLHOIST, $u_{PG} = u_H \mathfrak{p}(\theta_H)$ is nonfunctional. So, using the definition of \mathfrak{p} , u_H is not a variable and is neither \top nor \perp (condition 2 of BOOLHOIST).

By condition 4 of PGBOOLHOIST, $L_{PG}\theta_{PG}$ is not of the form $u_{PG}\theta_{PG} \approx \top$ or $u_{PG}\theta_{PG} \approx \perp$. Since $L_{PG}\theta_{PG} = L_H\theta_H$ and $u_{PG}\theta_{PG} = u_H\theta_H$, it follows that L_H is not of the form $u_H \approx \top$ or $u_H \approx \perp$ (condition 4 of BOOLHOIST).

Finally, the given inference ι_{PG} and the constructed inference ι_H are of the form (**) with $C_1[S_1] = C_H[S]$, $C'_2 = C_H \langle \perp \rangle \vee u_H \approx \top$, $S'_2 = S\sigma$, $\theta_1 = \theta_H$, and $\xi = \theta_{PG}$.

CASE 2: $p_{PG} = p_H.q$ for some nonempty q , u_H is not a variable but is variable-headed, and $u_H\theta$ is nonfunctional. Then we construct a FLUIDBOOLHOIST inference. By the assumption of this case, u_H is not a variable but is variable-headed (condition 1 of FLUIDBOOLHOIST).

Lemma 5.52 tells us that $s_{PG} = s_H \mathfrak{p}(\theta_H)$. Thus, the subterm of s_{PG} at position p_H is $u_H \mathfrak{p}(\theta_H)$. So q is a green position of $u_H \mathfrak{p}(\theta_H)$, and the subterm at that position is u_{PG} .

Since u_{PG} is the green subterm at position q of $u_H \mathfrak{p}(\theta_H)$, $u_H \mathfrak{p}(\theta_H) = (u_H \mathfrak{p}(\theta_H)) \langle u_{PG} \rangle_q$. Let $v = \lambda (u_H \mathfrak{p}(\theta_H)) \langle n \rangle_q$, where n is the appropriate De Bruijn index to refer to the initial λ .

Let z_H and x_H be fresh variables (condition 3 of FLUIDBOOLHOIST). We define $\theta'_H = (\theta_H[z_H \mapsto v\theta_{PG}, x_H \mapsto u_{PG}\theta_{PG}])$. Then $(z_H x_H)\theta'_H = v\theta_{PG}$ ($u_{PG}\theta_{PG} = (v u_{PG})\theta_{PG} = (v u_H \mathfrak{p}(\theta_H))\theta_{PG} = (u_H \mathfrak{p}(\theta_H)) \langle u_H \mathfrak{p}(\theta_H) \rangle_q \theta_{PG} u_H \mathfrak{p}(\theta_H) \theta_{PG} = u_H \mathfrak{p}(\theta_H) q(\theta_H) = u_H \theta_H u_H \theta'_H$). So θ'_H is a unifier of $z_H x_H$ and u_H . Thus, by definition of CSU (Definition 3.13), there exists a unifier $\sigma \in \text{CSU}(z_H x_H \equiv u_H)$ and a substitution ζ such that $x\sigma\zeta = x\theta'_H$ for all relevant variables x (condition 4 of FLUIDBOOLHOIST).

Since q is a green position of $u_H \mathfrak{p}(\theta_H)$, the type of $u_H \mathfrak{p}(\theta_H)$ and the type of $u_H \sigma$ is nonfunctional (condition 2 of FLUIDBOOLHOIST).

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

Condition 3 of PGBOOLHOIST states that u_{PG} is not a variable and is neither \top nor \perp . So $u_{PG}\theta_{PG} = x_H\sigma\zeta$ is neither \top nor \perp and thus $x_H\sigma$ is neither \top nor \perp (condition 7 of FLUIDBOOLHOIST). Moreover, $(z_H x_H)\sigma\zeta = v\theta_{PG}$ ($u_{PG}\theta_{PG} = (u_H \mathfrak{p}(\theta_H)) \langle u_{PG}\theta_{PG} \rangle_q \neq (u_H \mathfrak{p}(\theta_H)) \langle \perp \rangle_q \theta_{PG} = v\theta_{PG} \perp = (z_H \perp)\sigma\zeta$). Thus, $(z_H x_H)\sigma \neq (z_H \perp)\sigma$ (condition 5 of FLUIDBOOLHOIST).

The inferences ι_{PG} and ι_H are of the form (*) with $C_1[S_1] = C_H[S]$, $C_2[S_2] = (C_H \langle z_H \perp \rangle_{p_H} \vee x_H \approx \top[S])\sigma$, $\theta_1 = \theta_H$, $\theta_2 = \zeta$, $E = C_{PG} \langle \perp \rangle_{p_{PG}} \vee u_{PG} \approx \top$, and $\xi = \theta_{PG}$. In the following, we elaborate why $C_2\mathfrak{p}(\theta_2) = E\pi$ and $x\pi\mathfrak{q}(\theta_2) = x\xi$ for some substitution π and all variables x in E , as required for (*). The reasoning is similar to that in the FLUIDSUP case. We invoke Lemma 5.16 to obtain a substitution π such that

$\sigma\mathbf{p}(\zeta) = \mathbf{p}(\sigma\zeta)\pi$ and $\mathbf{q}(\sigma\zeta) = \pi\mathbf{q}(\zeta)$. Then

$$\begin{aligned}
C_2\mathbf{p}(\theta_2) &= (C_H \langle z_H \perp \rangle_{p_H} \vee x_H \approx \mathbf{T})\sigma\mathbf{p}(\zeta) \\
&= C_H\sigma\mathbf{p}(\zeta)\langle(z_H \perp)\sigma\mathbf{p}(\zeta)\rangle_{p_H} \vee x_H\sigma\mathbf{p}(\zeta) \approx \mathbf{T} \\
&\stackrel{(1)}{=} C_H\sigma\mathbf{p}(\zeta)\langle(z_H x_H)\sigma\mathbf{p}(\zeta)\langle\perp\rangle_q\rangle_{p_H} \vee u_{PG}\pi \approx \mathbf{T} \\
&\stackrel{(2)}{=} C_H\sigma\mathbf{p}(\zeta)\langle u_H\sigma\mathbf{p}(\zeta)\langle\perp\rangle_q\rangle_{p_H} \vee u_{PG}\pi \approx \mathbf{T} \\
&\stackrel{(3)}{=} C_H\sigma\mathbf{p}(\zeta)\langle\perp\rangle_{p_{PG}} \vee u_{PG}\pi \approx \mathbf{T} \\
&= C_H\mathbf{p}(\sigma\zeta)\langle\perp\rangle_{p_{PG}} \vee u_{PG}\pi \approx \mathbf{T} \\
&= (C_H\mathbf{p}(\sigma\zeta)\langle\perp\rangle_{p_{PG}} \vee u_{PG}\pi \approx \mathbf{T})\pi \\
&= (C_{PG}\langle\perp\rangle_{p_{PG}} \vee u_{PG}\pi \approx \mathbf{T})\pi \\
&= E\pi
\end{aligned}$$

Step (1) can be justified as follows: By Lemma 5.12, $z_H\sigma\mathbf{p}(\zeta)\mathbf{q}(\zeta) = v\theta_{PG}$. Since $z_H\sigma\mathbf{p}(\zeta)$ and v contain only nonfunctional variables, $z_H\sigma\mathbf{p}(\zeta)$ must be a λ -abstraction whose λ binds exactly one De Bruijn index, which is located at orange position $1.q$ w.r.t. $\downarrow_{\beta\eta\text{long}}$. Thus, $(z_H x_H)\sigma\mathbf{p}(\zeta)$ and $(z_H \perp)\sigma\mathbf{p}(\zeta)$ are identical up to the subterm at green position q , which is $x_H\sigma\mathbf{p}(\zeta)$ and \perp respectively. So, $(z_H \perp)\sigma\mathbf{p}(\zeta) = (z_H x_H)\sigma\mathbf{p}(\zeta)\langle\perp\rangle_q$. Moreover, since $(z_H x_H)\sigma\mathbf{p}(\zeta)\langle x_H\sigma\mathbf{p}(\zeta)\rangle_q = (z_H x_H)\sigma\mathbf{p}(\zeta) = u_H\sigma\mathbf{p}(\zeta) = u_H\mathbf{p}(\sigma\theta)\pi = u_H\mathbf{p}(\theta_H)\pi = u_H\mathbf{p}(\theta_H)\langle u_{PG}\rangle_q\pi$, we have $x_H\sigma\mathbf{p}(\zeta) = u_{PG}\pi$. For step (2), we use that $(z_H x_H)\sigma = u_H\sigma$. For step (3), we use that u_H is the green subterm of C_H at position p_H .

PGLOOBHOIST: Analogous to PGBOOLHOIST.

PGFALSEELIM: If ι_{PG} is a PGFALSEELIM inference

$$\frac{\overbrace{C'_{PG} \vee s_{PG} \approx t_{PG} \cdot \theta_{PG}}^{C_{PG}}}{C'_{PG} \cdot \theta_{PG}} \text{PGFALSEELIM}$$

then we construct a corresponding FALSEELIM inference ι_H . Let $C_H \cdot \theta_H = \mathcal{P}_{G(N)}^{-1}(C_{PG} \cdot \theta_{PG})$ and $C_H[S] = G_N^{-1}(C_H \cdot \theta_H)$. This clause $C_H[S]$ will be the premise of ι_H .

Condition 2 of PGFALSEELIM states that $s_{PG} \approx t_{PG}$ is strictly eligible in $C_{PG} \cdot \theta_{PG}$. Let $C_H = C'_H \vee s_H \approx t_H$, where $s_H \approx t_H$ is the literal that Lemma 5.52 guarantees to be strictly eligible w.r.t. any suitable σ (condition 2 of FALSEELIM), with $s_H\mathbf{p}(\theta_H) = s_{PG}$ and $t_H\mathbf{p}(\theta_H) = t_{PG}$.

Condition 1 of PGFALSEELIM states that $(s_{PG} \approx t_{PG})\theta_{PG} = \perp \approx \mathbf{T}$. So θ_H is a unifier of $s_H \equiv \perp$ and $t_H \equiv \mathbf{T}$. By construction, $S\theta_H$ is true. So there exists $(\sigma, U) \in \text{CSU}^{\text{upto}}(S, s_H \equiv \perp, t_H \equiv \mathbf{T})$ (condition 1 of FALSEELIM) and a substitution ζ such that $U\zeta$ is true and $x\sigma\zeta = x\theta_H$ for all variables x in $C_H[S]$.

The inferences ι_{PG} and ι_H are of the form (**) with $C_1[S_1] = C_H[S]$, $C'_2 = C'_H \vee s_H \approx t_H$, $S'_2 = U$, $\theta_1 = \theta_H$, and $\xi = \theta_{PG}$.

PGARGCONG: If ι_{PG} is a PGARGCONG inference

$$\frac{\overbrace{C'_{PG} \vee s_{PG} \approx s'_{PG} \cdot \theta_{PG}}^{C_{PG}}}{C'_{PG} \vee s_{PG} \text{diff}(\tau, v)(u_{PG}, v_{PG}) \approx s'_{PG} \text{diff}(\tau, v)(u_{PG}, v_{PG}) \cdot \theta_{PG}} \text{PGARGCONG}$$

then we construct a corresponding ARGCONG inference ι_H . Let $C_H \cdot \theta_H = \mathcal{P}_{\mathcal{G}(N)}^{-1}(C_{PG} \cdot \theta_{PG})$ and $C_H[S] = \mathcal{G}_N^{-1}(C_H \cdot \theta_H)$. This clause $C_H[S]$ will be the premise of ι_H .

Condition 3 of PGARGCONG states that $s_{PG} \approx s'_{PG}$ is strictly eligible in $C_{PG} \cdot \theta_{PG}$. Let $C_H = C'_H \vee s_H \approx s'_H$, where $s_H \approx s'_H$ is the literal that Lemma 5.52 guarantees to be strictly eligible w.r.t. any suitable σ (condition 2 of ARGCONG), with $s_H p(\theta_H) = s_{PG}$ and $s'_H p(\theta_H) = s'_{PG}$.

Let x be a fresh variable (condition 3 of ARGCONG). Let $\theta'_H = \theta_H[x \mapsto \text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG})]$.

Condition 1 of PGARGCONG states that s_{PG} is of type $\tau \rightarrow v$. Since PG does not use type variables, τ and v are ground types, and thus $s_{PG} \theta_{PG} = s_H \theta_H = s_H \theta'_H$ is of type $\tau \rightarrow v$. Let σ be the most general type substitution such that $s_H \sigma$ is of functional type (condition 1 of ARGCONG), and let ζ be a substitution such that $y \sigma \zeta = y \theta'_H$ for all type and term variables y in $C_H[S]$.

Then, the given inference ι_{PG} and the constructed inference ι_H are of the form (**) with $C_1[S_1] = C_H[S]$, $C'_2 = C'_H \sigma \vee s_H \sigma x \approx s'_H \sigma x$, $S'_2 = S \sigma$, $\theta_1 = \theta'_H$, $\xi = q(\theta'_H)$, $\zeta = \theta'_H$, and $\sigma = \{\}$.

PGEXT: If ι_{PG} is a PGEXT inference

$$\frac{C_{PG} \langle u_{PG} \rangle \cdot \theta_{PG}}{C_{PG} \langle v_{PG} \rangle \vee u_{PG} \text{ diff}\langle\tau, v\rangle(u_{PG}, v_{PG}) \not\approx v_{PG} \text{ diff}\langle\tau, v\rangle(u_{PG}, v_{PG}) \cdot \rho} \text{ PGEXT}$$

then we construct a corresponding EXT or FLUIDEXT inference ι_H . Let $C_H \cdot \theta_H = \mathcal{P}_{\mathcal{G}(N)}^{-1}(C_{PG} \cdot \theta_{PG})$ and $C_H[S] = \mathcal{G}_N^{-1}(C_H \cdot \theta_H)$. This clause $C_H[S]$ will be the premise of ι_H . Let $L_{PG}.s_{PG}.p_{PG}$ be the position of the green subterm u_{PG} in C_{PG} . Condition 1 of PGEXT states that $L_{PG}.s_{PG}.p_{PG}$ is eligible in $C_{PG} \cdot \theta_{PG}$. Let $L_H.s_H.p_H$ be the position in C_H that Lemma 5.52 guarantees to be eligible in $C_H[S]$ w.r.t. any suitable σ (condition 3 of EXT or condition 7 of FLUIDEXT). Let u_H be the subterm of C_H at position $L_H.s_H.p_H$. By Lemma 5.52, one of the following cases applies.

CASE 1: $p_{PG} = p_H$. Then we construct an EXT inference.

Condition 2 of PGEXT states that $u_{PG} = u_H p(\theta_H)$ is of functional type. Let σ be the most general type substitution such that $u_H \sigma$ is of type $\tau \rightarrow v$ for some types τ and v (condition 1 of EXT). Let y be a fresh variable of the same type as $u_H \sigma$ (condition 2 of EXT). Let $\theta'_H = \theta_H[y \mapsto v_{PG} \rho]$. Let ζ be a substitution such that $x \sigma \zeta = x \theta'_H$ for all type and term variables x in $C_H[S]$ and for $x = y$.

By condition 3 of PGEXT, $v_{PG} \in T_{PG}$ is a term whose nonfunctional yellow subterms are different fresh variables. Then v_{PG} and $y p(\theta_H)$ are equal up to renaming of variables because p replaces the nonfunctional yellow subterms of $v_{PG} \theta_H$ by distinct fresh variables. Since the purpose of this proof is to show that ι_{PG} is redundant, a property that is independent of the variable names in ι_{PG} 's conclusion, we can assume without loss of generality that $v_{PG} = y p(\theta_H)$ and $\theta_{PG} = q(\theta'_H)$. Then the given inference ι_{PG} and the constructed inference ι_H are of the form (**) with $C_1[S_1] = C_H[S]$, $C'_2 = C_H \sigma \langle y \rangle \vee u_H \sigma \text{ diff}\langle\tau, v\rangle(u_H \sigma, y) \not\approx y \text{ diff}\langle\tau, v\rangle(u_H \sigma, y)$, $S'_2 = S \sigma$, $\theta_1 = \theta'_H$, $\xi = \rho$, $\zeta = \theta'_H$, and $\sigma = \{\}$.

CASE 2: $p_{PG} = p_H.q$ for some nonempty q , u_H is not a variable but is variable-headed, and $u_H \theta$ is nonfunctional. Then we construct a FLUIDEXT inference.

Lemma 5.52 tells us that $s_{PG} = s_H p(\theta_H)$. Thus, the subterm of s_{PG} at position p_H is $u_H p(\theta_H)$. So q is a green position of $u_H p(\theta_H)$, and the subterm at that position is u_{PG} —i.e.,

$u_H p(\theta_H) = (u_H p(\theta_H)) \langle u_{PG} \rangle_q$. Let $t = \lambda (u_H p(\theta_H)) \langle n \rangle_q$, where n is the appropriate De Bruijn index to refer to the initial λ .

We define $\theta'_H = \theta_H[x \mapsto u_{PG}\theta_{PG}, y \mapsto v_{PG}\rho, z \mapsto t\theta_{PG}]$. Then $(z x)\theta'_H = (t u_{PG})\theta_{PG} = u_H p(\theta_H)\theta_{PG} = u_H p(\theta_H)q(\theta_H) = u_H\theta_H = u_H\theta'_H$. So θ'_H is a unifier of $z x$ and u_H . Thus, by definition of CSU (Definition 3.13), there exists a unifier $\sigma \in \text{CSU}(z x \equiv u_H)$ and a substitution ζ such that $x\sigma\zeta = x\theta'_H$ for all relevant variables x (condition 4 of FLUIDEXT).

By the assumption of this case, u_H is not a variable but is variable-headed (condition 1 of FLUIDEXT) and $u_H\theta$ is nonfunctional (condition 2 of FLUIDEXT).

By condition 4 of PGEXT, $u_{PG}\theta_{PG} \neq v_{PG}\rho$. Thus, $(z x)\sigma\zeta = (z x)\theta'_H = (t u_{PG})\theta_{PG} = (u_H p(\theta_H)) \langle u_{PG}\theta_{PG} \rangle_q \neq (u_H p(\theta_H)) \langle v_{PG}\rho \rangle_q = (z y)\theta'_H = (z y)\sigma\zeta$. So, $(z x)\sigma \neq (z y)\sigma$ (condition 5 of FLUIDEXT).

Since $z\sigma\zeta = t\theta_{PG}$ and $t\theta_{PG} \neq \lambda 0$ because q is nonempty, we have $z\sigma \neq \lambda 0$ (condition 6 of FLUIDEXT).

By condition 3 of PGEXT, $v_{PG} \in T_{PG}$ is a term whose nonfunctional yellow subterms are different fresh variables. Then v_{PG} and $y\mathfrak{p}(\theta'_H)$ are equal up to renaming of variables because \mathfrak{p} replaces the nonfunctional yellow subterms of $v_{PG}\rho$ by distinct fresh variables. As above, we can assume without loss of generality that $y\mathfrak{p}(\theta'_H) = v_{PG}$ and $w\rho = wq(\theta'_H)$ for all variables w in C_{PG} and in v_{PG} , using the fact that ρ coincides with θ_{PG} on all variables in C_{PG} .

The inferences ι_{PG} and ι_H are of the form $(*)$ with $C_1[\![S_1]\!] = C_H[\![S]\!]$, $C_2[\![S_2]\!] = (C_H \langle zy \rangle_{p_H} \vee x\text{diff}\langle\alpha, \beta\rangle(x, y) \not\approx y\text{diff}\langle\alpha, \beta\rangle(x, y))\sigma\mathfrak{p}(\zeta)$, $\theta_1 = \theta_H$, $\theta_2 = \zeta$, $E = C_{PG} \langle v_{PG} \rangle_{p_{PG}} \vee u_{PG}\text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG}) \not\approx v_{PG}\text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG})$, $\xi = \rho$. In the following, we elaborate why $C_2\mathfrak{p}(\theta_2) = E\pi$ and $x\pi q(\theta_2) = x\xi$ for some substitution π and all variables x in E , as required for $(*)$. The reasoning is similar to that in the FLUIDSUP case. We invoke Lemma 5.16 to obtain a substitution π such that $\sigma\mathfrak{p}(\zeta) = \mathfrak{p}(\sigma\zeta)\pi$ and $q(\sigma\zeta) = \pi q(\zeta)$. Then

$$\begin{aligned} C_2\mathfrak{p}(\theta_2) &= (C_H \langle z y \rangle_{p_H} \vee x\text{diff}\langle\alpha, \beta\rangle(x, y) \not\approx y\text{diff}\langle\alpha, \beta\rangle(x, y))\sigma\mathfrak{p}(\zeta) \\ &= C_H\sigma\mathfrak{p}(\zeta) \langle (z y)\sigma\mathfrak{p}(\zeta) \rangle_{p_H} \vee (x\text{diff}\langle\alpha, \beta\rangle(x, y) \not\approx y\text{diff}\langle\alpha, \beta\rangle(x, y))\sigma\mathfrak{p}(\zeta) \\ &\stackrel{(1)}{=} (C_H\sigma\mathfrak{p}(\zeta) \langle (z x)\sigma\mathfrak{p}(\zeta) \langle y\sigma\mathfrak{p}(\zeta) \rangle_q \rangle_{p_H} \vee (x\text{diff}\langle\alpha, \beta\rangle(x, y) \not\approx y\text{diff}\langle\alpha, \beta\rangle(x, y))\sigma\mathfrak{p}(\zeta)) \\ &\stackrel{(2)}{=} (C_H\sigma\mathfrak{p}(\zeta) \langle u_H\sigma\mathfrak{p}(\zeta) \langle y\sigma\mathfrak{p}(\zeta) \rangle_q \rangle_{p_H} \vee (x\text{diff}\langle\alpha, \beta\rangle(x, y) \not\approx y\text{diff}\langle\alpha, \beta\rangle(x, y))\sigma\mathfrak{p}(\zeta)) \\ &\stackrel{(3)}{=} C_H\sigma\mathfrak{p}(\zeta) \langle y\sigma\mathfrak{p}(\zeta) \rangle_{p_{PG}} \vee (x\text{diff}\langle\alpha, \beta\rangle(x, y) \not\approx y\text{diff}\langle\alpha, \beta\rangle(x, y))\sigma\mathfrak{p}(\zeta) \\ &\stackrel{(4)}{=} C_H\sigma\mathfrak{p}(\zeta) \langle v_{PG} \rangle_{p_{PG}} \vee (u_{PG}\text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG}) \not\approx v_{PG}\text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG}))\pi \\ &= (C_H\mathfrak{p}(\sigma\zeta) \langle v_{PG} \rangle_{p_{PG}} \vee u_{PG}\text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG}) \not\approx v_{PG}\text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG}))\pi \\ &= (C_{PG} \langle v_{PG} \rangle_{p_{PG}} \vee u_{PG}\text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG}) \not\approx v_{PG}\text{diff}\langle\tau, v\rangle(u_{PG}, v_{PG}))\pi \\ &= E\pi \end{aligned}$$

Step (1) can be justified as follows: By Lemma 5.12, $z\sigma\mathfrak{p}(\zeta)q(\zeta) = t\theta_{PG}$. Since $z\sigma\mathfrak{p}(\zeta)$ and t contain only nonfunctional variables, $z\sigma\mathfrak{p}(\zeta)$ must be a λ -abstraction whose λ binds exactly one De Bruijn index, which is located at orange position $1.q$ w.r.t. $\downarrow_{\beta\eta\text{long}}$. Thus, $(z x)\sigma\mathfrak{p}(\zeta)$ and $(z y)\sigma\mathfrak{p}(\zeta)$ are identical up to the subterm at green position q , which is $x\sigma\mathfrak{p}(\zeta)$ and $y\sigma\mathfrak{p}(\zeta)$ respectively. So, $(z y)\sigma\mathfrak{p}(\zeta) = (z x)\sigma\mathfrak{p}(\zeta) \langle y\sigma\mathfrak{p}(\zeta) \rangle_q$.

For step (2), we use that $(z x)\sigma = u_H\sigma$. For step (3), we use that u_H is the green subterm of C_H at position p_H . For step (4), we use that $x\sigma\mathfrak{p}(\zeta) = x\mathfrak{p}(\sigma\zeta)\pi = x\mathfrak{p}(\theta'_H)\pi = u_{PG}\pi$ and similarly $y\sigma\mathfrak{p}(\zeta) = v_{PG}\pi$.

PGDIFF: If ι_{PG} is a PGDIFF inference

$$\frac{u \mathbf{diff} \langle \tau, v \rangle(u, v) \not\approx u \mathbf{diff} \langle \tau, v \rangle(u, v) \vee u s \approx v s \cdot \theta_{\text{PG}}}{\text{PGDIFF}}$$

then we use the following DIFF inference ι_H :

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

Let θ_H be a grounding substitution with $\alpha\theta_H = \tau$, $\beta\theta_H = v$, $y\theta_H = u\theta_{\text{PG}}$, and $z\theta_H = v\theta_{\text{PG}}$. By condition 2 of PGDIFF, $u, v, s \in T_{\text{PG}}$ are terms whose nonfunctional yellow subterms are different fresh variables. Then u and $y\mathbf{p}(\theta_H)$ are equal up to renaming of variables because \mathbf{p} replaces the nonfunctional yellow subterms of $u\theta_{\text{PG}}$ by distinct fresh variables. The same holds for v and $z\mathbf{p}(\theta_H)$ and for s and $x\mathbf{p}(\theta_H)$. Since the purpose of this proof is to show that ι_{PG} is redundant, a property that is independent of the variable names in ι_{PG} 's conclusion, we can assume without loss of generality that $u = y\mathbf{p}(\theta_H)$, $v = z\mathbf{p}(\theta_H)$, $s = x\mathbf{p}(\theta_H)$, and $\theta_{\text{PG}} = \mathbf{q}(\theta_H)$. Then $C_H\mathbf{p}(\theta_H) = C_{\text{PG}}$ and $w\mathbf{q}(\theta_H) = w\theta_{\text{PG}}$ for all variables w in C_{PG} . So ι_{PG} and ι_H have the form $(*)$ with $C_1 = C_H$, $E = C_{\text{PG}}$, $\xi = \theta_{\text{PG}}$, $\pi = \{\}$, and $\theta_1 = \theta_H$. \square

5.5. Trust and Simple Redundancy. In this subsection, we define a notion of trust for each level and connect them. Ultimately, we prove that simple redundancy $(H\text{Red}_C^*, H\text{Red}_I^*)$ as defined in Section 3.7 implies redundancy $(H\text{Red}_C, H\text{Red}_I)$ as defined in Section 5.4.4.

5.5.1. First-Order Level. In this subsubsection, let \succ be an admissible term order for PFinf (Definition 5.20).

Definition 5.54 (Trust). A closure $C \cdot \theta \in \mathcal{C}_{\text{PF}}$ trusts a closure $D \cdot \rho \in N \subseteq \mathcal{C}_{\text{PF}}$ if the variables in D can be split into two sets X and Y such that

1. for every literal $L \in D$ containing a variable $x \in X$, there exists a variable z in a literal $K \in C$ such that $x\rho$ is a subterm of $z\theta$ and $L\rho \preceq K\theta$; and
2. for all grounding substitutions ρ' with $x\rho' = x\rho$ for all $x \notin Y$, we have $D \cdot \rho' \in N$.

Lemma 5.55. Let R be a confluent term rewrite system oriented by \succ whose only Boolean normal forms are \top and \perp . If a closure $C \cdot \theta \in \mathcal{C}_{\text{PF}}$ trusts a closure $D \cdot \rho \in N \subseteq \mathcal{C}_{\text{PF}}$ and $C \cdot \theta$ is variable-irreducible, then there exists a closure $D \cdot \rho' \in \text{irred}_R(N)$ with $D \cdot \rho' \preceq D \cdot \rho$ such that $R \cup \{D \cdot \rho'\} \models_{\text{o}\lambda} D \cdot \rho$.

Proof. Let X and Y the sets from Definition 5.54. We define a substitution ρ' by $y\rho' = y\rho \downarrow_R$ for all variables $y \in Y$ and $x\rho' = x\rho$ for all variables $x \notin Y$. By condition 2 of the definition of trust, we have $D \cdot \rho' \in N$. Moreover, $D \cdot \rho' \preceq D \cdot \rho$ by (O2)_{PF} because R is oriented by \succ .

We show that $D \cdot \rho'$ is also variable-irreducible. If a variable y is in Y , then $y\rho'$ is reduced w.r.t. R by definition of ρ' . If a variable x is in X , then consider an occurrence of x in a literal $L \cdot \rho' \in D \cdot \rho'$. We must show that $x\rho' = x\rho$ is irreducible w.r.t. the rules in R smaller than $L \cdot \rho'$. By condition 1, there exists a variable z in a literal $K \in C$ such that $x\rho$ is a subterm of $z\theta$ and $L\rho \preceq K\theta$. Since $C \cdot \theta$ and thus $K \cdot \theta$ is variable-irreducible w.r.t. R , $z\theta$ is irreducible w.r.t. the rules in R smaller than $K \cdot \theta$. Since $L\rho' \preceq L\rho \preceq K\theta$ and $x\rho = x\rho'$ is a subterm of $z\theta$, this implies that $x\rho'$ is irreducible w.r.t. the rules in R smaller than $L \cdot \rho'$.

For the Boolean condition of variable-irreducibility, we use that R 's only Boolean normal forms are \top and \perp .

To show that $R \cup \{D \cdot \rho'\} \models D \cdot \rho$, it suffices to prove that $x\rho \rightarrow_R^* x\rho'$ for all x in D . If x is in X then $x\rho = x\rho'$ and hence $x\rho \rightarrow_R^* x\rho'$. Otherwise, x is in Y and $x\rho' = x\rho \downarrow_R$ and thus $x\rho \rightarrow_R^* x\rho'$. \square

5.5.2. Indexed Partly Substituted Ground Higher-Order Level. In this subsubsection, let \succ be an admissible term order for $IPGInf$ (Definition 5.21).

The definition of trust is almost identical to the corresponding definition on the PF level. However, we need to take into account that PF level subterms correspond to yellow subterms on the IPG level.

Definition 5.56 (Trust). A closure $C \cdot \theta \in \mathcal{C}_{IPG}$ trusts a closure $D \cdot \rho \in N \subseteq \mathcal{C}_{IPG}$ if the variables in D can be split into two sets X and Y such that

1. for every literal $L \in D$ containing a variable $x \in X$, there exists a variable z in a literal $K \in C$ such that $x\rho$ is a yellow subterm of $z\theta$ and $L\rho \preceq K\theta$; and
2. for all grounding substitutions ρ' with $x\rho' = x\rho$ for all $x \notin Y$, we have $D \cdot \rho' \in N$.

Lemma 5.57. *If a closure $C \cdot \theta \in \mathcal{C}_{IPG}$ trusts a closure $D \cdot \rho \in N \subseteq \mathcal{C}_{IPG}$ w.r.t. \succ , then $\mathcal{F}(C \cdot \theta)$ trusts $\mathcal{F}(D \cdot \rho) \in \mathcal{F}(N)$ w.r.t. $\succ_{\mathcal{F}}$.*

Proof. Assume $C \cdot \theta$ trusts $D \cdot \rho$. Thus, the variables of D can be split into two sets X and Y such that conditions 1 and 2 of the definition of trust are satisfied. Note that the variables of D and $\mathcal{F}(D)$ coincide. We claim that $\mathcal{F}(C \cdot \theta)$ trusts $\mathcal{F}(D \cdot \rho)$ via the variable sets X and Y .

Let $\mathcal{F}(L) \in \mathcal{F}(D)$ be a literal containing a variable $x \in X$. Then $L \in D$ contains x as well. By the definition of trust (Definition 5.56), there exists a variable z in some literal $K \in C$ such that $x\rho$ is a yellow subterm of $z\theta$ and $L\rho \preceq K\theta$. It holds that $\mathcal{F}(L) \in \mathcal{F}(D)$ is a literal containing $x \in X$. Since $x\rho$ is a yellow subterm of $z\theta$ we have that $\mathcal{F}(x\rho)$ is a subterm of $\mathcal{F}(z\theta)$. Moreover, we have $\mathcal{F}(L\rho) \preceq_{\mathcal{F}} \mathcal{F}(K\theta)$. Hence, condition 1 of Definition 5.54 is satisfied.

It remains to show that also condition 2 of Definition 5.54 holds—i.e., that for any grounding substitution ρ' with $x\rho' = x\mathcal{F}(\rho)$ for all $x \notin Y$, we have $\mathcal{F}(D \cdot \rho') \in \mathcal{F}(N)$. We define a substitution $\rho'' : x \mapsto \mathcal{F}^{-1}(\rho'(x))$. Since \mathcal{F}^{-1} maps ground terms to ground terms, ρ'' is grounding. Moreover, since \mathcal{F} is bijective on ground terms, $x\rho'' = \mathcal{F}^{-1}(\rho'(x)) = \mathcal{F}^{-1}(x\mathcal{F}(\rho)) = x\rho$ for all $x \notin Y$. By the definition of trust (Definition 5.56), $D \cdot \rho'' \in N$ and therefore $\mathcal{F}(D \cdot \rho') = \mathcal{F}(D \cdot \rho'') \in \mathcal{F}(N)$. \square

Lemma 5.58. *Let R be a confluent term rewrite system on T_{PF} oriented by $\succ_{\mathcal{F}}$ whose only Boolean normal forms are \top and \perp . If a closure $C \cdot \theta \in \mathcal{C}_{IPG}$ trusts a closure $D \cdot \rho \in N \subseteq \mathcal{C}_{IPG}$ and $C \cdot \theta$ is variable-irreducible, then there exists a closure $D \cdot \rho' \in \text{irred}_R(N)$ with $\mathcal{F}(D \cdot \rho') \preceq_{\mathcal{F}} \mathcal{F}(D \cdot \rho)$ such that $R \cup \{\mathcal{F}(D \cdot \rho')\} \models_{o\lambda} \mathcal{F}(D \cdot \rho)$.*

Proof. By Lemma 5.57, $\mathcal{F}(C \cdot \theta)$ trusts $\mathcal{F}(D \cdot \rho) \in \mathcal{F}(N)$. By Lemma 5.55, there exists a closure $D_0 \cdot \rho'_0 \in \text{irred}_R(\mathcal{F}(N)) = \mathcal{F}(\text{irred}_R(N))$ with $D_0 \cdot \rho'_0 \preceq_{\mathcal{F}} \mathcal{F}(D \cdot \rho)$ such that $R \cup \{D_0 \cdot \rho'_0\} \models_{o\lambda} \mathcal{F}(D \cdot \rho)$. Thus, there must exist a closure $D \cdot \rho' \in \text{irred}_R(N)$ with $\mathcal{F}(D \cdot \rho') \preceq_{\mathcal{F}} \mathcal{F}(D \cdot \rho)$ such that $R \cup \{\mathcal{F}(D \cdot \rho')\} \models_{o\lambda} \mathcal{F}(D \cdot \rho)$. \square

5.5.3. Partly Substituted Ground Higher-Order Level. In this subsubsection, let \succ be an admissible term order for $PGInf$ (Definition 5.22).

As terms can also contain parameters we also need to account for this case in the definition of trust for the PG level. Specifically, we must add a condition that the variables in Y may not appear in parameters in the trusted closure.

Definition 5.59 (Trust). A closure $C \cdot \theta \in \mathcal{C}_{PG}$ trusts a closure $D \cdot \rho \in N \subseteq \mathcal{C}_{PG}$ if the variables in D can be partitioned into two sets X and Y such that

1. for every literal $L \in D$ containing a variable $x \in X$ outside of parameters, there exists a literal $K \in C$ containing a variable z outside of parameters such that $x\rho$ is a yellow subterm of $z\theta$ and $L\rho \preceq K\theta$; and
2. all variables in Y do not appear in parameters in D and for all grounding substitutions ρ' with $x\rho' = x\rho$ for all $x \notin Y$, $D \cdot \rho' \in N$.

Lemma 5.60. *If a closure $C \cdot \theta \in \mathcal{C}_{PG}$ trusts a closure $D \cdot \rho \in N \subseteq \mathcal{C}_{PG}$ w.r.t. \succ , then $\mathcal{J}(C \cdot \theta)$ trusts $\mathcal{J}(D \cdot \rho) \in \mathcal{J}(N)$ w.r.t. $\succ_{\mathcal{J}}$.*

Proof. Let $C \cdot \theta \in \mathcal{C}_{PG}$ and $D \cdot \rho \in \mathcal{C}_{PG}$ such that $C \cdot \theta$ trusts $D \cdot \rho$. Thus, the variables of D can be partitioned into two sets X and Y such that conditions 1 and 2 of the definition of trust are satisfied. Let X' be the set X without the variables that only occur in parameters in D . We claim that $\mathcal{J}(C \cdot \theta)$ trusts $\mathcal{J}(D \cdot \rho)$ using the sets X' and Y that split the variables in $\mathcal{J}(D \cdot \rho)$.

Since the parameters disappear after the application of the transformation \mathcal{J} , the set $X' \cup Y$ contains all variables occurring in $\mathcal{J}(D \cdot \rho)$. As all the other variables are preserved by \mathcal{J} we have that X' and Y are partition of the variables in $\mathcal{J}(D \cdot \rho)$.

Let $\mathcal{J}_\rho(L) \in \mathcal{J}_\rho(D)$ be a literal containing a variable $x \in X$. Then $L \in D$ contains $x \in X' \subseteq X$ outside of parameters. By Definition 5.59, there exists a literal $K \in C$ containing a variable z outside of parameters such that $x\rho$ is a yellow subterm of $z\theta$ and $L\rho \preceq K\theta$. It holds that $\mathcal{J}_\rho(K) \in \mathcal{J}_\rho(D)$ is a literal containing $x \in X$. Since $x\rho$ is a yellow subterm of $z\theta$, we have that $\mathcal{J}(x\rho)$ is a yellow subterm of $\mathcal{J}(z\theta)$, since \mathcal{J} preserves yellow subterms. Then, by Lemma 5.9, $\mathcal{J}_\rho(x)\mathcal{J}(\rho)$ is a yellow subterm of $\mathcal{J}_\theta(z)\mathcal{J}(\theta)$. Moreover, we have $\mathcal{J}_\rho(K)\mathcal{J}(\rho) = \mathcal{J}(K\rho) \preceq_{\mathcal{J}} \mathcal{J}(L\theta) = \mathcal{J}_\rho(L)\mathcal{J}(\theta)$ again using Lemma 5.9. Hence, condition 1 of Definition 5.56 is satisfied.

Let ρ' be a grounding substitution on the IPG level with $x\rho' = x\mathcal{J}(\rho)$ for all $x \notin Y$. Since $\mathcal{J}(D \cdot \rho) = \mathcal{J}_\rho(D) \cdot \mathcal{J}(\rho)$, we must show that $\mathcal{J}_\rho(D) \cdot \rho' \in \mathcal{J}(N)$. Since ρ' is a grounding substitution and \mathcal{J} is a bijection on ground terms, there exists a substitution $\rho'' = \mathcal{J}^{-1}(\rho')$. This substitution ρ'' is grounding and $x\rho'' = \mathcal{J}^{-1}(x\rho') = \mathcal{J}^{-1}(\mathcal{J}(x\rho)) = x\rho$ for all $x \notin Y$. Since $C \cdot \theta$ trusts $D \cdot \rho$, we have $D \cdot \rho'' \in N$ and thus $\mathcal{J}_{\rho''}(D) \cdot I(\rho'') = \mathcal{J}(D \cdot \rho'') \in \mathcal{J}(N)$. Since the variables in Y do not occur in parameters in D , we have $\mathcal{J}_{\rho''}(D) = \mathcal{J}_\rho(D)$. Moreover, $\mathcal{J}(\rho'') = \rho'$. Hence, condition 2 of Definition 5.56 is satisfied. \square

Lemma 5.61. *Let R be a confluent term rewrite system on \mathcal{T}_{PF} oriented by $\succ_{\mathcal{J}F}$ whose only Boolean normal forms are \mathbf{T} and \mathbf{L} . If a closure $C \cdot \theta \in \mathcal{C}_{PG}$ trusts a closure $D \cdot \rho \in N \subseteq \mathcal{C}_{PG}$ and $C \cdot \theta$ is variable-irreducible, then there exists a closure $D \cdot \rho' \in \text{irred}_R(N)$ with $\mathcal{F}(\mathcal{J}(D \cdot \rho')) \preceq_{\mathcal{J}F} \mathcal{F}(\mathcal{J}(D \cdot \rho))$ such that $R \cup \{\mathcal{F}(\mathcal{J}(D \cdot \rho'))\} \models_{o\lambda} \mathcal{F}(\mathcal{J}(D \cdot \rho))$.*

Proof. By Lemma 5.60, $\mathcal{J}(C \cdot \theta)$ trusts $\mathcal{J}(D \cdot \rho) \in \mathcal{J}(N)$. By Lemma 5.58, there exists a closure $D_0 \cdot \rho'_0 \in \text{irred}_R(\mathcal{J}(N)) = \mathcal{J}(\text{irred}_R(N))$ with $\mathcal{F}(D_0 \cdot \rho'_0) \preceq \mathcal{F}(\mathcal{J}(D \cdot \rho))$ such that $R \cup \{\mathcal{F}(D_0 \cdot \rho'_0)\} \models_{o\lambda} \mathcal{F}(\mathcal{J}(D \cdot \rho))$. Thus, there must exist a closure $D \cdot \rho' \in \text{irred}_R(N)$ with $\mathcal{F}(\mathcal{J}(D \cdot \rho')) \preceq \mathcal{F}(\mathcal{J}(D \cdot \rho))$ such that $R \cup \{\mathcal{F}(\mathcal{J}(D \cdot \rho'))\} \models_{o\lambda} \mathcal{F}(\mathcal{J}(D \cdot \rho))$. \square

5.5.4. Full Higher-Order Level. In this subsubsection, let \succ be an admissible term order (Definition 3.16), extended to be an admissible term order for $PGInf$ as in Section 5.4.4. We have defined trust for level H in Definition 3.27.

Lemma 5.62. *Let $C[S] \in C_H$ and $D[T] \in N \subseteq C_H$. Let $C\theta \in \text{Gnd}(C[S])$ and $D\rho \in \text{Gnd}(D[T])$. If the θ -instance of $C[S]$ trusts the ρ -instance of $D[T]$, then $\mathcal{P}(C \cdot \theta)$ trusts $\mathcal{P}(D \cdot \rho) \in \mathcal{PG}(N)$.*

Proof. Let X and Y be a partition of the variables in D such that the variables in X fulfill condition (i) and the variables in Y fulfill condition (ii). Let X' be the set of variables occurring in $\mathcal{P}(D \cdot \rho)$ originating from $x\mathbf{p}(\rho)$ for some $x \in X$. Define the set Y' analogously. We claim that $\mathcal{P}(C \cdot \theta)$ trusts $\mathcal{P}(D \cdot \rho)$ using the sets X' and Y' . It holds that X' and Y' are a partition of the variables in $\mathcal{P}(D \cdot \rho)$.

REGARDING X' : We need to show that for every literal $L' \in D\mathbf{p}(\rho)$ containing a variable $x' \in X'$ outside of parameters, there exists a literal $K' \in C\mathbf{p}(\theta)$ containing a variable z' outside of parameters such that $x'\mathbf{q}(\rho)$ is a yellow subterm of $z'\mathbf{q}(\theta)$ and $L'\mathbf{q}(\rho) \preceq K'\mathbf{q}(\theta)$.

Any literal $L' \in D\mathbf{p}(\rho)$ containing a variable $x' \in X'$ outside of parameters must originate from a literal $L \in D$ containing a variable $x \in X$ outside of parameters, where $L' = L\mathbf{p}(\rho)$ and x' occurs in $x\mathbf{p}(\rho)$. By condition (i) of Definition 3.27, this implies that there exists a literal $K \in C$ and a substitution σ such that $z\theta = z\sigma\rho$ for all variables z in C and $L \preceq K\sigma$.

By (O10) with the substitution $\mathbf{p}(\rho)$, since x' occurs outside of parameters of $L' = L\mathbf{p}(\rho) = L\mathbf{p}(\rho)$, it also occurs outside of parameters of $K\sigma\mathbf{p}(\rho)$.

By Lemma 5.16, there exists a substitution π such that $\sigma\mathbf{p}(\rho) = \mathbf{p}(\sigma\rho)\pi$ and $\mathbf{q}(\sigma\rho) = \pi\mathbf{q}(\rho)$. So $K\sigma\mathbf{p}(\rho) = K\mathbf{p}(\sigma\rho)\pi = K\mathbf{p}(\theta)\pi$ and x' occurs outside of parameters in this literal. Since $K\mathbf{p}(\theta)$ contains only nonfunctional variables, there exists a variable z' occurring outside of parameters in $K\mathbf{p}(\theta)$ such that x' is a yellow subterm of $z'\pi$. Thus, $x'\mathbf{q}(\rho)$ is a yellow subterm of $z'\pi\mathbf{q}(\rho) = z'\mathbf{q}(\sigma\rho) = z'\mathbf{q}(\theta)$, which is what we needed to show.

REGARDING Y' : We must show that for all grounding substitutions ρ' with $x\rho' = x\mathbf{q}(\rho)$ for all $x \notin Y'$, we have $D\mathbf{p}(\rho) \cdot \rho' \in \mathcal{PG}(N)$.

Let ρ' be a substitution with $x\rho' = x\mathbf{q}(\rho)$ for all $x \notin Y'$. Then, for all variables $x \notin Y$, we have $x\mathbf{p}(\rho)\rho' = x\mathbf{q}(\rho)\mathbf{q}(\rho) = x\rho$ by Lemma 5.12. By condition (ii) of Definition 3.27, the variables in Y do not appear in the constraints T of $D[T]$. So, $T\mathbf{p}(\rho)\rho' = T\rho$ and thus $\mathbf{p}(\rho)\rho'$ is true. Therefore, $D \cdot \mathbf{p}(\rho)\rho' \in \mathcal{G}(N)$ and $\mathcal{P}(D \cdot \mathbf{p}(\rho)\rho') \in \mathcal{PG}(N)$. By Lemma 5.13 and 5.14, $\mathcal{P}(D \cdot \mathbf{p}(\rho)\rho') = D\mathbf{p}(\mathbf{p}(\rho)\rho') \cdot \mathbf{q}(\mathbf{p}(\rho)\rho') = D\mathbf{p}(\rho) \cdot \rho'$ because $x\rho' = x\mathbf{q}(\rho)$ for all $x \notin Y'$ and in particular for all x not introduced by $\mathbf{p}(\rho)$. Therefore, $D\mathbf{p}(\rho) \cdot \rho' \in \mathcal{PG}(N)$.

A variable in $y\mathbf{p}(\rho)$ can occur in a parameter in $\mathcal{P}(D \cdot \rho)$ only if the variable y occurs in a parameter in D . Since all variables in Y do not appear in parameters, the variables in Y' do not appear in parameters either. \square

Lemma 5.63. *Let R be a confluent term rewrite system on \mathcal{T}_{PF} oriented by $\succ_{\mathcal{IF}}$ whose only Boolean normal forms are \top and \perp . Let $C[S] \in C_H$ and $D[T] \in N \subseteq C_H$. Let $C\theta \in \text{Gnd}(C[S])$ and $D\rho \in \text{Gnd}(D[T])$. If the θ -instance of $C[S]$ trusts the ρ -instance of $D[T]$ and $C \cdot \theta$ is variable-irreducible, then there exists a closure $D \cdot \rho' \in \text{irred}_R(\mathcal{G}(N))$ with $\mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho'))) \preceq_{\mathcal{IF}} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$ such that $R \cup \{\mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho')))\} \models_{o\lambda} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$.*

Proof. By Lemma 5.62, $\mathcal{I}(C \cdot \theta)$ trusts $\mathcal{P}(D \cdot \rho) \in \mathcal{PG}(N)$. By Lemma 5.61, there exists a closure $D_0 \cdot \rho'_0 \in \text{irred}_R(\mathcal{P}(N)) = \mathcal{P}(\text{irred}_R(N))$ with $\mathcal{F}(\mathcal{I}(D_0 \cdot \rho'_0)) \preceq_{\mathcal{IF}} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$ such that $R \cup \{\mathcal{F}(\mathcal{I}(D_0 \cdot \rho'_0))\} \models_{o\lambda} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$. Thus, there must exist a closure

$D \cdot \rho' \in \text{irred}_R(N)$ with $\mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho'))) \preceq_{\mathcal{J}\mathcal{F}} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$ such that $R \cup \{\mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho')))\} \models_{\text{o}\lambda} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$. \square

Lemma 5.64. *Let $N \subseteq \mathcal{C}_H$. Then $H\text{Red}_C^*(N) \subseteq H\text{Red}_C(N)$.*

Proof. Let $C[\![S]\!] \in H\text{Red}_C^*(N)$. Let R be a confluent term rewrite system on \mathcal{T}_F oriented by $\succ_{\mathcal{J}\mathcal{F}}$ whose only Boolean normal forms are \top and \perp . Let $C' \cdot \theta' \in \text{irred}_R(\mathcal{FI}\mathcal{P}(C[\![S]\!]))$, i.e., there exists some $C\theta \in \text{Gnd}(C[\![S]\!])$ such that $\mathcal{FI}\mathcal{P}(C \cdot \theta) = C' \cdot \theta'$. By Lemma 5.17, $\mathcal{F}(C\theta) = C'\theta'$.

We make a case distinction on which condition of Definition 3.28 applies to $C\theta \in \text{Gnd}(C[\![S]\!])$.

CONDITION 1: There exist an indexing set I and for each $i \in I$ a ground instance $D_i\rho_i$ of a clause $D_i[\![T_i]\!] \in N$, such that

- (a) $\mathcal{F}(\{D_i\rho_i \mid i \in I\}) \models_{\text{o}\lambda} \mathcal{F}(C\theta)$;
- (b) for all $i \in I$, $D_i\rho_i \prec C\theta$; and
- (c) for all $i \in I$, the θ -instance of $C[\![S]\!]$ trusts the ρ_i -instance of $D_i[\![T_i]\!]$.

We show that $C[\![S]\!] \in H\text{Red}_C(N)$ by condition 1 of Definition 5.46, i.e., we show that

$$R \cup \{E \in \text{irred}_R(\mathcal{FI}\mathcal{P}(N)) \mid E \prec_{\mathcal{J}\mathcal{F}} C'\theta'\} \models_{\text{o}\lambda} C'\theta'$$

By Lemma 5.63 and point (c) above, there exists a closure $D'_i \cdot \rho'_i \in \text{irred}_R(\mathcal{FI}\mathcal{P}(N))$ with $D'_i \cdot \rho'_i \preceq_{\mathcal{J}\mathcal{F}} \mathcal{F}(\mathcal{I}(\mathcal{P}(D_i \cdot \rho_i)))$ such that $R \cup \{D'_i \cdot \rho'_i\} \models_{\text{o}\lambda} \mathcal{F}(\mathcal{I}(\mathcal{P}(D_i \cdot \rho_i)))$. By Lemma 5.17, $D'_i \rho'_i \preceq_{\mathcal{J}\mathcal{F}} \mathcal{F}(D_i \rho_i)$ and $R \cup \{D'_i \rho'_i\} \models_{\text{o}\lambda} \mathcal{F}(D_i \rho_i)$. With point (a) above, it follows that

$$R \cup \{D'_i \rho'_i \mid i \in I\} \models_{\text{o}\lambda} \mathcal{F}(C\theta)$$

It remains to show that $D'_i \rho'_i \prec_{\mathcal{J}\mathcal{F}} \mathcal{F}(C\theta)$ for all $i \in I$. Since $D'_i \rho'_i \preceq_{\mathcal{J}\mathcal{F}} \mathcal{F}(D_i \rho_i)$, it suffices to show that $\mathcal{F}(D_i \rho_i) \prec_{\mathcal{J}\mathcal{F}} \mathcal{F}(C\theta)$. This follows from point (b) above, Lemma 5.17, and Definitions 5.28 and 5.37.

CONDITION 2: There exists a ground instance $D\rho$ of some $D[\![T]\!] \in N$ such that

- (a) $D\rho = C\theta$;
- (b) $C[\![S]\!] \sqsupset D[\![T]\!]$; and
- (c) the θ -instance of $C[\![S]\!]$ trusts the ρ -instance of $D[\![T]\!]$.

By Lemma 5.63 and point (c) above, there exists a closure $D' \cdot \rho' \in \text{irred}_R(\mathcal{FI}\mathcal{P}(N))$ with $D' \cdot \rho' \preceq_{\mathcal{J}\mathcal{F}} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$ such that $R \cup \{D' \cdot \rho'\} \models_{\text{o}\lambda} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$.

We distinguish two subcases.

CASE 1: $D' \cdot \rho' = \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$. Then we can show that $C[\![S]\!] \in H\text{Red}_C(N)$ by condition 2 of Definition 5.46, using points (a) and (b) above and Lemma 5.17.

CASE 2: $D' \cdot \rho' \prec_{\mathcal{J}\mathcal{F}} \mathcal{F}(\mathcal{I}(\mathcal{P}(D \cdot \rho)))$. Then we show that $C[\![S]\!] \in H\text{Red}_C(N)$ by condition 1 of Definition 5.46—i.e.,

$$R \cup \{E \in \text{irred}_R(\mathcal{FI}\mathcal{P}(N)) \mid E \prec_{\mathcal{J}\mathcal{F}} C'\theta'\} \models_{\text{o}\lambda} C'\theta'$$

Since $C'\theta' = \mathcal{F}(C\theta) = \mathcal{F}(D\rho) = \mathcal{FI}\mathcal{P}(D \cdot \rho)$ by point (a) above and Lemma 5.17, it suffices to show

$$R \cup \{E \in \text{irred}_R(\mathcal{FI}\mathcal{P}(N)) \mid E \prec_{\mathcal{J}\mathcal{F}} \mathcal{FI}\mathcal{P}(D \cdot \rho)\} \models_{\text{o}\lambda} \mathcal{FI}\mathcal{P}(D \cdot \rho)$$

This follows directly from the three defining properties of $D' \cdot \rho'$ above. \square

Lemma 5.65. *Let $N \subseteq \mathcal{C}_H$. Then $H\text{Red}_I^*(N) \subseteq H\text{Red}_I(N)$.*

Proof. Let $\iota \in HRed_I^*(N)$. Let $C_1[S_1], \dots, C_m[S_m]$ be its premises and $C_{m+1}[S_{m+1}]$ its conclusion. Let $\theta_1, \dots, \theta_{m+1}$ be a tuple of substitutions for which ι is rooted in $FInf$ (Definition 3.31). Since $\iota \in HRed_I^*(N)$, by Definition 3.32, there exists an indexing set I and for each $i \in I$ a ground instance $D_i\rho_i$ of a clause $D_i[T_i] \in N$, such that

1. $\mathcal{F}(\{D_i\rho_i \mid i \in I\}) \models_{o\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$;
2. ι is a DIFF inference or for all $i \in I$, $D_i\rho_i \prec C_m\theta_m$; and
3. for all $i \in I$, the θ_{m+1} -instance of $C_{m+1}[S_{m+1}]$ trusts the ρ_i -instance of $D_i[T_i]$.

By Definition 5.47, we must show that for all confluent term rewrite systems R oriented by $\succ_{\mathcal{IF}}$ whose only Boolean normal forms are \top and \perp such that $C_{m+1} \cdot \theta_{m+1}$ is variable-irreducible, we have

$$R \cup O \models_{o\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$$

where $O = \text{irred}_R(\mathcal{FIPG}(N))$ if ι is a DIFF inference and $O = \{E \in \text{irred}_R(\mathcal{FIPG}(N)) \mid E \prec_{\mathcal{IF}} \mathcal{F}(C_m\theta_m)\}$ otherwise.

By Lemma 5.63 and point 3 above, there exists a closure $D'_i \cdot \rho'_i \in \text{irred}_R(\mathcal{FIPG}(N))$ with $D'_i \cdot \rho'_i \preceq_{\mathcal{IF}} \mathcal{F}(\mathcal{I}(\mathcal{P}(D_i \cdot \rho_i)))$ such that $R \cup \{D'_i \cdot \rho'_i\} \models_{o\lambda} \mathcal{F}(\mathcal{I}(\mathcal{P}(D_i \cdot \rho_i)))$. By Lemma 5.17, $D'_i \rho'_i \preceq_{\mathcal{IF}} \mathcal{F}(D_i \rho_i)$ and $R \cup \{D'_i \rho'_i\} \models_{o\lambda} \mathcal{F}(D_i \rho_i)$. With point 1 above, it follows that

$$R \cup \{D'_i \rho'_i \mid i \in I\} \models_{o\lambda} \mathcal{F}(C_{m+1}\theta_{m+1})$$

If ι is a DIFF inference, we are done. For the other inferences, it remains to show that $D'_i \rho'_i \prec_{\mathcal{IF}} \mathcal{F}(C_m\theta_m)$ for all $i \in I$. Since $D'_i \rho'_i \preceq_{\mathcal{IF}} \mathcal{F}(D_i \rho_i)$, it suffices to show that $\mathcal{F}(D_i \rho_i) \prec_{\mathcal{IF}} \mathcal{F}(C_m\theta_m)$. This follows from point 2 above, Lemma 5.17, and Definitions 5.28 and 5.37. \square

5.6. 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 5.4, we prove a completeness property for each of the calculi that roughly states the following. For any saturated set N_∞ that does not contain an empty closure, there exists a term rewrite system R and a corresponding interpretation \mathcal{I} such that \mathcal{I} is a model of the closures in N_∞ that are variable-irreducible w.r.t. R .

Moreover, to prepare the eventual extension the model of the variable-irreducible instances to all instances, for each level, we show a property that roughly states the following: For any set of closures N_0 that contains all closures of the form $C \cdot \rho$ for all ρ whenever it contains a closure $C \cdot \theta$, then the variable-irreducible instances of N_0 entail all of N_0 .

Finally, in level H, we bring everything together by showing that the constructed model is also a model of the variable-irreducible ground instances of N_0 and thus of N_0 itself. It follows that the calculus $HInf$ is refutationally complete.

5.6.1. First-Order Levels. In this subsubsection, let \succ be an admissible term order for $PFinf$ (Definition 5.20), and let $pfsel$ be a selection function on \mathcal{CPF} (Definition 5.18).

The completeness proof for $PFinf$ 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 5.66 (R_N). Let N be a set of ground first-order closures with $\perp \notin T(N)$. 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:

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

- 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) there exists a closure $C_0 \cdot \theta \in N$ such that $C = C_0 \theta$;
- (CC2) $C_0 \cdot \theta$ is variable-irreducible w.r.t. R_C ;
- (CC3) $C = C' \vee s \approx t$ for some clause C' and terms s and t ;
- (CC4) s is nonfunctional;
- (CC5) the root of s is not a logical symbol;
- (CC6) if t is Boolean, then $t = \top$
- (CC7) $s \succ t$;
- (CC8) $s \approx t$ is maximal in C ;
- (CC9) there are no selected literals in $C_0 \cdot \theta$;
- (CC10) s is irreducible by R_C ;
- (CC11) $R_C \not\models_{\text{fol}} C$;
- (CC12) $R_C \cup \{s \rightarrow t\} \not\models_{\text{fol}} C'$.

In this case, we say that $C_0 \cdot \theta$ produces $s \rightarrow t$ and that $C_0 \cdot \theta$ 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 5.67. *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)_{PF}.

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 $e, e' \in T_F \cup C_F$. Without

loss, we assume $e \succ e'$. Inspecting the rules of Definition 5.66, it follows that $s \succeq s'$. By the subterm property (O3)_{PF}, 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 5.66. \square

Lemma 5.68. *The normal form of any ground Boolean term w.r.t. R_N is \top or \perp .*

Proof. Inspecting the rules of Definition 5.66, in particular (CC5), 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 5.67, $s \succ s'$. By the induction hypothesis, s' reduces to \top or \perp . Therefore, s reduces to \top or \perp . \square

Lemma 5.69. *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 5.67, 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 5.66, 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 5.70. *If a closure $C_0 = C'_0 \vee s_0 \approx t_0 \cdot \theta \in \mathcal{C}_{\text{PF}}$ produces $s_0 \theta \rightarrow t_0 \theta$, then $R_N \not\models_{\text{fol}} C'_0 \theta$.*

Proof. Let $C = C_0 \theta$, $C' = C'_0 \theta$, $s = s_0 \theta$, and $t = t_0 \theta$. By (CC7) and (CC8), all terms in C are smaller or equal to s . By (CC12), 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 5.66, in particular the irreducibility conditions, and because R_N is confluent and terminating (Lemma 5.67). So, $R_C \cup \{s \rightarrow t\} \not\models_{\text{fol}} C'$ implies $R_N \not\models_{\text{fol}} C'$. \square

Lemma 5.71. *If $C \cdot \theta \in \mathcal{C}_{\text{PF}}$ is productive, then it is variable-irreducible w.r.t. R_N .*

Proof. Let $s \rightarrow t$ be the rule produced by $C \cdot \theta$. By (CC2), $C \cdot \theta$ is variable-irreducible w.r.t. R_C . Let $s' \rightarrow t' \in R_N \setminus R_C$. Then $s' \approx t' \in \Delta_e$ for some $e \in \mathcal{T}_F \cup \mathcal{C}_F$ that is larger than $C\theta$. So if e is a term, then $s' = e \succ s$ and thus $s' \approx t' \succ s \approx t$. If e is a clause, then its maximal literal (which is $s' \approx t'$ by $(\Delta 2)$, $(\Delta 3)$, and (CC8)) is at least as large as C 's maximal literal (which is $s \approx t$ by (CC8)). So in either case, $s' \approx t' \succeq s \approx t$. Since $s \approx t$ is the maximal literal of $C\theta$, $s' \approx t'$ is at least as large as each literal of $C\theta$. So the rule $s' \approx t'$ has no effect on the variable-irreducibility of $C \cdot \theta$ by Definition 5.24. Therefore, $C \cdot \theta$ is variable-irreducible w.r.t. R_N . \square

Lemma 5.72. *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 .

If the rewrite position is in a proper subterm of $\mathcal{F}(u)$, by definition of \mathcal{F} , the rewrite position corresponds to a proper yellow subterm of u . Yellow subterms of a functional term remain when applying the term to an argument. So the same rewrite step can be applied to obtain $\mathcal{F}(u \text{ diff}_{s,t}^{\tau,v}) \rightarrow_{R_N} \mathcal{F}(v \text{ diff}_{s,t}^{\tau,v})$.

For a rewrite in the root of the term, the rewrite rule must originate from $(\Delta 3)$ because the terms are functional. One of the conditions of $(\Delta 3)$ then yields the claim. \square

Lemma 5.73. *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 5.72 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 $(O5)_{PF}$, 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 5.74. *R_N is a $\models_{o\lambda}$ -interpretation.*

Proof. We must prove all conditions listed in Section 3.7.

- By Lemma 5.68, the Boolean type has exactly two elements, namely the interpretations of \top and \perp . The rule $(\Delta 1)$ ensures that the symbols $\neg, \wedge, \vee, \rightarrow, \approx^\tau, \not\approx^\tau$ are interpreted as the corresponding logical operations. Note that R_s never contains any rules rewriting s because s is smaller than any clause containing s . So s can be reducible w.r.t. R_s only when one of its proper subterms is reducible. Since every term has a normal form, adding rules only for the irreducible terms is sufficient.
- By Lemma 5.11, we have $\mathcal{F}(\mathcal{I}(u) \text{ diff}_{s,t}^{\tau,v}) = \mathcal{F}(u \text{ diff } \langle \tau, v \rangle(s, t))$ for all $u, s, t \in \mathcal{T}_{\text{ground}}(\Sigma_H)$. Since \mathcal{I} is a bijection on ground terms, Lemma 5.73 proves the extensionality condition in Section 3.7.
- The argument congruence condition in Section 3.7 follows from Lemma 5.72 in the same way. \square

Lemma 5.75. *If the premises of an inference are variable-irreducible w.r.t. a ground rewrite system R with $R \subseteq \succ$ and the inference is not a PFEXT or PFDIFF inference, then the conclusion is also variable-irreducible w.r.t. R .*

Proof. Let $C_0 \cdot \theta$ be the conclusion of the inference. By Definition 5.24, we have to show that for all literals $L \cdot \theta$ of $C_0 \cdot \theta$ and all variables x of L , $x\theta$ is irreducible w.r.t. all rules $l \rightarrow r \in R$ with $L\theta \succ l \approx r$ and all Boolean subterms of $x\theta$ are either \top or \perp . Since the premises of the inference are variable-irreducible, this is evident for all literals that occur also in one of the premises. The Boolean subterm condition is satisfied for all inferences because no inference other than PFEXT and PFDIFF introduces a variable that is not

present in the premises. It remains to check the irreducibility condition for newly introduced literals in $C_0 \cdot \theta$.

For PFSUP inferences, the only newly introduced literal has the form $L[t'] \cdot \theta$, where $L[u] \cdot \theta$ is a literal of the second premise and $(t \approx t') \cdot \theta$ is a literal in the first premise with $t\theta = u\theta$ and $t\theta \succ t'\theta$. Let x be a variable in $L[t']$. If x occurs in t' , then $x\theta$ is irreducible w.r.t. all rules $l \rightarrow r \in R$ with $t\theta \approx t'\theta \succ l \approx r$ since the first premise is variable-irreducible w.r.t. R . On the other hand, $x\theta$ must also be irreducible w.r.t. all rules $l \rightarrow r \in R$ with $t\theta \approx t'\theta \preceq l \approx r$, since then $l \succeq t\theta \succ t'\theta \succeq x\theta$. Therefore, $x\theta$ is irreducible w.r.t. all rules $l \rightarrow r \in R$. If x occurs in $L[t']$ but not in t' , then it occurs in $L[u]$, and because the second premise is variable-irreducible w.r.t. R , $x\theta$ is irreducible w.r.t. all rules $l \rightarrow r \in R$ with $L[u]\theta \succ l \approx r$. Since $L[t']\theta \prec L[u]\theta$, $x\theta$ is also irreducible w.r.t. all rules $l \rightarrow r \in R$ with $L[t']\theta \succ l \approx r$.

For all other inferences, it is easy to verify that whenever a variable x occurs in a newly introduced literal $L \cdot \theta$ in the conclusion, then x occurs also in a literal $L' \cdot \theta$ in the premise with $L\theta \preceq L'\theta$, so the premise's variable-irreducibility implies the conclusion's variable-irreducibility. \square

We employ a variant of Bachmair and Ganzinger's framework of reducing counter-examples [2, Sect. 4.2]. Let $N \subseteq C_{\text{PF}}$ with $\perp \notin T(N)$. A closure $C_0 \cdot \theta \in C_{\text{PF}}$ is called a *counterexample* if it is variable-irreducible w.r.t. R_N and $R_N \not\models_{\text{fol}} C_0 \cdot \theta$. An inference reduces a counterexample $C_0 \cdot \theta$ if its main premise is $C_0 \cdot \theta$, its side premises are in N and true in R_N , and its conclusion is a counterexample smaller than $C_0 \cdot \theta$. An inference system has the *reduction property for counterexamples* if for all $N \subseteq C_{\text{PF}}$ and all counterexamples $C_0 \cdot \theta \in N$, there exists an inference from N that reduces $C_0 \cdot \theta$.

Lemma 5.76. *Let $C_0 \cdot \theta \in N$ be a counterexample. Let L_0 be a literal in $C_0 \cdot \theta$ that is eligible and negative or strictly eligible and positive. Let $C = C_0\theta$ and $L = L_0\theta$. We assume that the larger side of L is reducible by a rule $s \rightarrow s' \in R_C$. Then the inference system PFInf reduces the counterexample $C_0 \cdot \theta$.*

Proof. Let p be the position of C that is located at the larger side of L and reducible by $s \rightarrow s'$.

First, we claim that p is not at or below a variable position of C_0 and thus $s = s_0\theta$ for a subterm s_0 of C_0 that is not a variable.

To see this, assume for a contradiction that p is at or below a variable position of C_0 , i.e., there is a variable x in C such that s is a subterm of $x\theta$.

If s were Boolean, then s must be \top or \perp by variable-irreducibility, contradicting the fact that $s \rightarrow s' \in R_C$ because Definition 5.66 does not produce rules rewriting \top or \perp .

So s is not a Boolean term. Then, by the rules of Definition 5.66, $s \rightarrow s' \in R_C$ implies that $C \succeq s \approx s'$. By variable-irreducibility, since s is a subterm of $x\theta$, $L \preceq s \approx s'$, where L is the literal of C containing position p . Since L contains s , it follows that L must be positive and its larger side must be s . Since p is eligible and s is not a Boolean term, L must be the maximal literal of C . So, since $C \succeq s \approx s'$ and $L \preceq s \approx s'$, we have $L = s \approx s'$. But this contradicts this lemma's assumption that $R_N \not\models_{\text{fol}} C$.

This concludes the proof of our claim that $s = s_0\theta$ for a subterm s_0 of C_0 that is not a variable.

If the subterm s at position p is not a green subterm of C , then it must be contained in functional green subterm t of C . Let q be the position of t in C . Since p is eligible in

$C_0 \cdot \theta$, q is also eligible in $C_0 \cdot \theta$. Let t' the normal form of t w.r.t. R_N . Let $u = \mathcal{F}^{-1}(t)$ and $v = \mathcal{F}^{-1}(t')$. Then $\mathcal{F}(u) \leftrightarrow_{R_N}^* \mathcal{F}(v)$. By Lemma 5.72, $\mathcal{F}(u \text{ diff}_{u,v}^{\tau,v}) \leftrightarrow_{R_N}^* \mathcal{F}(v \text{ diff}_{u,v}^{\tau,v})$. Since t contains s , it is reducible w.r.t. R_N , and thus $\mathcal{F}(u) = t \succ t' = \mathcal{F}(v)$. Thus, we can apply PFEXT to reduce the counterexample. To fulfill the conditions of PFEXT on v , we must replace the nonfunctional yellow subterms of v by fresh variables and choose ρ accordingly. 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)_{PF} and because $\mathcal{F}(u) \succ \mathcal{F}(v)$.

It remains to show that the conclusion is variable-irreducible. First, consider one of the fresh variables introduced to replace the nonfunctional yellow subterms of v . Since yellow subterms in v correspond to subterms in t' , and t' is a normal form, $x\rho$ is irreducible w.r.t. R_N . Next, consider a variable x that occurs in a literal $L \cdot \theta$ in the conclusion but is not one of the fresh variables. Then x occurs also in a literal $L' \cdot \theta$ in the premise $C_0 \cdot \theta$ with $L\theta \preceq L'\theta$. Since $C_0 \cdot \theta$ is variable-irreducible, this shows that the conclusion is variable-irreducible.

Otherwise, s is a green subterm of C . Then we make a case distinction on which case of Definition 5.66 the rule $s \rightarrow s'$ originates from:

- (Δ1) Then the root of s is a logical symbol and $s \notin \{\top, \perp\}$. By Lemma 5.68, R_N reduces s to \top or to \perp . By our claim above, s_0 is not a variable and since $s = s_0\theta$, s_0 has a logical symbol at its root.
- 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 PFCLAUSIFY is applicable to $C_0 \cdot \theta$ and the conclusion of this inference is smaller than it. Moreover, the conclusion is equivalent to $C_0 \cdot \theta$ by Lemma 5.74 and variable-irreducible by Lemma 5.75 .
- Otherwise, we apply either PFBOOLHOIST (if s_0 reduces to \perp) or PFLOOBHOIST (if s_0 reduces to \top). In both cases, the conclusion of the inference is smaller than $C_0 \cdot \theta$. Moreover, the conclusion is equivalent to $C_0 \cdot \theta$ by Lemma 5.74 and variable-irreducible by Lemma 5.75 .
- (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 PFBOOLHOIST to reduce the counterexample, again using Lemma 5.74 and Lemma 5.75 .
- (Δ3) Then s is functional and reducible w.r.t. R_N .Then we can proceed as in the PFEXT case above, using s in the role of t .
- (Δ4) Then some closure $D_0 \vee t \approx t' \cdot \rho$ with $(D_0 \vee t \approx t')\rho = D \vee s \approx s'$ smaller than C produces the rule $s \rightarrow s'$. We claim that the counterexample C is reduced by the inference

$$\frac{D_0 \vee t \approx t' \cdot \rho \quad C_0 \langle s_0 \rangle \cdot \theta}{D_0 \vee C_0 \langle t' \rangle \cdot (\rho \cup \theta)} \text{PFSUP}$$

This superposition is a valid inference:

- $t\rho = s = s_0\theta$.
- By our claim above, s_0 is not a variable.
- s_0 is nonfunctional by (CC4).
- We have $s \succ s'$ by (CC7).
- $D \vee s \approx s' \prec C[s]$ because $D \vee s \approx s'$ produces a rule in R_C .
- The position p of s in $C_0 \cdot \theta$ is eligible by assumption of this lemma.

- The literal $t \approx t'$ is eligible in $(D_0 \vee t \approx t') \cdot \rho$ by (CC8) and (CC9). 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 (CC12).
- If $t' \rho$ is Boolean, then $t' \rho = \top$ by (CC6).

As $D_0 \vee t \approx t' \cdot \rho$ is productive, $R_N \not\models_{\text{fol}} D$ by Lemma 5.70. Hence $D \vee C[s']$ is equivalent to $C[s']$, which is equivalent to $C[s]$ w.r.t. R_N . Moreover, $(D_0 \vee t \approx t') \cdot \rho$ is variable-irreducible by Lemma 5.71. So $D \vee C[s']$ is variable-irreducible by Lemma 5.75. It remains to show that the new counterexample $D \vee C[s']$ is strictly smaller than C . Using (O2)_{PF}, $C[s'] \prec C$ 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 5.77. *The inference system PFInf has the reduction property for counterexamples.*

Proof. Let $C_0 \cdot \theta \in N$ be a counterexample—i.e., a closure in $\text{irred}_{R_N}(N)$ that is false in R_N . We must show that there is an inference from N that reduces $C_0 \cdot \theta$; i.e., the inference has main premise $C_0 \cdot \theta$, side premises in N that are true in R_N , and a conclusion that is a smaller counterexample than $C_0 \cdot \theta$. For all claims of a reducing inference in this proof, we use Lemma 5.75 to show that the conclusion is variable-irreducible.

Let L_0 be an eligible literal in $C_0 \cdot \theta$. Let $C = C_0 \theta$. We proceed by a case distinction:

CASE 1: $L_0 \theta$ is of the form $s \not\approx s'$.

- Case 1.1: $s = s'$. Then PFEQRES 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 5.69. 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 5.76.

CASE 2: $L_0 \theta$ 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_0 is eligible, but not strictly eligible. Then $L_0 \theta$ occurs more than once in C . So we can apply PFEQFACT to reduce the counterexample.
- Case 2.2: L_0 is strictly eligible and s is reducible by R_C . Then we apply Lemma 5.76.
- Case 2.3: L_0 is strictly eligible and $s = \perp$. Then, since $s \succ s'$, we have $s' = \top$ by (O4)_{PF}. So, PFFALSEELIM reduces the counterexample.
- Case 2.4: L_0 is strictly eligible and s is functional. Then we apply PFARGCONG to reduce the counterexample. The conclusion is smaller than the premise by (O5)_{PF}. By Lemma 5.73, there must be at least one choice of u and v in the PFARGCONG rule such that the conclusion is a counterexample.
- Case 2.5: L_0 is strictly eligible and $s \neq \perp$ is nonfunctional and not reducible by R_C . Since $R_N \not\models_{\text{fol}} C$, $C_0 \cdot \theta$ cannot be productive. So at least one of the conditions of $(\Delta 4)$ of Definition 5.66 is violated. (CC1), (CC2), (CC3), (CC4), (CC7), (CC10), and (CC11) are clearly satisfied.

For (CC5), (CC6), (CC8), and (CC9), we argue as follows:

- (CC5): 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 5.68 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 .
- (CC6): If s' were a Boolean other than \top , since $s \succ s'$, we would have $s \neq \top, \perp$ by (O4)_{PF}. Moreover, $s' \succeq \perp$, and thus $C \succeq s \approx \perp$. Since s is not reducible by R_C , is is

also irreducible by $R_{s \approx \perp} \subseteq R_C$. So (Δ2) 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.

- (CC8): By (CC6), L_0 cannot be selected and thus eligibility implies maximality.
- (CC9): By (CC6), L_0 cannot be selected. If another literal was selected, L_0 would not be eligible.

So (CC12) must be violated. Then $R_C \cup \{s \rightarrow s'\} \models_{\text{fol}} C'$, where C' is the subclause of C with $L_0\theta$ 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 (CC8), $r \preceq s$. 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 PFEQFACT reduces the counterexample. \square

Using Lemma 5.77 and the same ideas as for Theorem 4.9 of Bachmair and Ganzinger’s framework [2], we obtain the following theorem:

Theorem 5.78. *Let N be a set of closures that is saturated up to redundancy w.r.t. PFI_{nf} and PFRed_{I} , and N does not contain $\perp \cdot \theta$ for any θ . Then $R_N \models_{\text{o}\lambda} \text{irred}_{R_N}(N)$.*

Proof. By Lemma 5.74, it suffices to show that $R_N \models_{\text{fol}} \text{irred}_{R_N}(N)$. For a proof by contradiction, we assume that $R_N \not\models_{\text{fol}} \text{irred}_{R_N}(N)$. Then N contains a minimal counterexample, i.e., a closure $C_0 \cdot \theta$ that is variable-irreducible w.r.t. R_N with $R_N \not\models_{\text{fol}} C_0 \cdot \theta$. Since PFI_{nf} has the reduction property for counterexamples by Lemma 5.77, there exists an inference that reduces $C_0 \cdot \theta$ —i.e., an inference ι with main premise $C_0 \cdot \theta$, side premises in N that are true in R_N , and a conclusion $\text{concl}(\iota)$ that is smaller than $C_0 \cdot \theta$, variable-irreducible w.r.t. R_N , and false in R_N . By saturation up to redundancy, $\iota \in \text{PFRed}_{\text{I}}$. By Definition 5.26, we have $R_N \cup \{E \in \text{irred}_{R_N}(N) \mid E \prec C_0 \cdot \theta\} \models_{\text{o}\lambda} \text{concl}(\iota)$. By minimality of the counterexample $C_0 \cdot \theta$, the closures $\{E \in \text{irred}_{R_N}(N) \mid E \prec C_0 \cdot \theta\}$ must be true in R_N , and it follows that $\text{concl}(\iota)$ is true in R_N , a contradiction. \square

Lemma 5.79. *Let R be a confluent term rewrite system oriented by \succ whose only Boolean normal forms are \top and \perp . Let $N \subseteq \mathcal{C}_{\text{PF}}$ such that for every $C \cdot \theta \in N$ and every grounding substitution ρ that coincides with θ on all variables not occurring in C , we have $C \cdot \rho \in N$. Then $R \cup \text{irred}_R(N) \models_{\text{o}\lambda} N$.*

Proof. Let $C \cdot \theta \in N$. We must show that $R \cup \text{irred}_R(N) \models_{\text{o}\lambda} C \cdot \theta$. We define a substitution θ' by $x\theta' = (x\theta)\downarrow_R$ for variables x occurring in C and $x\theta' = x\theta$ for all other variables. Then $R \cup \{C \cdot \theta'\} \models_{\text{o}\lambda} C \cdot \theta$. Moreover, θ' is grounding and coincides with θ on all variables not occurring in C . By the assumption of this lemma, we have $C \cdot \theta' \in N$. Finally, we observe that the closure $C \cdot \theta'$ is variable-irreducible w.r.t. R —i.e., $C \cdot \theta' \in \text{irred}_R(N)$. It follows that $R \cup \text{irred}_R(N) \models_{\text{o}\lambda} C \cdot \theta$. \square

5.6.2. Indexed Partly Substituted Ground Higher-Order Level.

In this subsubsection, let \succ be an admissible term order for IPGInf (Definition 5.21), let ipgsel be a selection function on \mathcal{C}_{IPG} , and let $N \subseteq \mathcal{C}_{\text{IPG}}$ such that N is saturated up to redundancy w.r.t. IPGInf and $\perp \cdot \theta \notin N$ for all θ . We write R for the term rewrite system

$R_{\mathcal{F}(N)}$ constructed in the previous subsubsection w.r.t. $\succ_{\mathcal{F}}$ and $\mathcal{F}(\text{ipgsel})$. We write $t \sim s$ for $\mathcal{F}(t) \leftrightarrow_R^* \mathcal{F}(s)$, where $t, s \in \mathcal{T}_{\text{ground}}(\Sigma_I)$.

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 5.80 (Argument congruence). *Let $s \sim s'$ for $s, s' \in \mathcal{T}_{\text{ground}}(\Sigma_I)$. Let $u \in \mathcal{T}_{\text{ground}}(\Sigma_I)$. Then $s u \sim s' u$.*

Proof. Let t, t', v be terms and θ a grounding substitution such that $t\theta = s$, $t'\theta = s'$, $v\theta = u$, and the nonfunctional yellow subterms of t, t', v are different variables. Let ρ the substitution resulting from R -normalizing all values of θ (via \mathcal{F}). Then there exists the inference

$$\frac{}{t \text{ diff}_{t\rho, t'\rho}^{\tau, v} \not\approx t' \text{ diff}_{t\rho, t'\rho}^{\tau, v} \vee t v \approx t' v \cdot \rho} \text{IPGDIFF}$$

which we call ι . By construction of ρ , its conclusion is variable-irreducible.

Since N is saturated, ι is redundant and thus $R \cup \mathcal{F}(\text{irred}_R(N)) \models \mathcal{F}(\text{concl}(\iota))$. Hence $R \models \mathcal{F}(\text{concl}(\iota))$ by Theorem 5.78 and Lemma 5.36.

We have $t\rho \sim t\theta = s \sim s' = t'\theta \sim t'\rho$. By Lemma 5.72, $R \models \mathcal{F}(t\rho \text{ diff}_{t\rho, t'\rho}^{\tau, v}) \approx \mathcal{F}(t'\rho \text{ diff}_{t\rho, t'\rho}^{\tau, v})$. Using Lemma 5.5, it follows that $R \models \mathcal{F}((t v \approx t' v)\rho)$. Since applying a functional term to an argument preserves nonfunctional yellow subterms of both the functional term and its argument, we have $s u = (t v)\theta \sim (t v)\rho$ and $s' u = (t' v)\theta \sim (t' v)\rho$. So $R \models \mathcal{F}(s u \approx s' u)$ and thus $s u \sim s' u$. \square

The following lemma and its proof are essentially identical to Lemma 54 of Bentkamp et al. [6]. 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 5.81. *Let $s \in \mathcal{T}(\Sigma_I)$, 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 $s \in \mathcal{T}^\lambda(\Sigma_I)$. Here, for $t_1, t_2 \in \mathcal{T}_{\text{ground}}^\lambda(\Sigma_I)$, 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} [12] 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 König's lemma [21], 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:

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

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 $f(\bar{\tau})\bar{t}$ 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 $s\sigma = (f(\bar{\tau})\bar{t})\sigma$. 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 5.80 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 5.73, 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 $n_3(s) = S(s) = 1 + S(u)$ because s is nonground. Moreover, $S(u) = S(v) = n_3(v)$. 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 $\mathcal{J}^{\text{IPG}} = (\mathcal{U}^{\text{IPG}}, \mathcal{J}^{\text{IPG}}, \mathcal{I}^{\text{IPG}}, \mathcal{L}^{\text{IPG}})$. The construction proceeds as in the completeness proof of the original λ -superposition calculus [6]. 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}^{IPG} 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
- $\mathcal{E}_{\tau \rightarrow v}(\llbracket \mathcal{F}(s) \rrbracket_R) \in \mathcal{D}_{\tau \rightarrow v}$ 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 $\llbracket \mathcal{F}(s) \rrbracket_R = \llbracket \mathcal{F}(t) \rrbracket_R$ and $\mathcal{E}_\tau(\llbracket \mathcal{F}(u) \rrbracket_R) = \mathcal{E}_\tau(\llbracket \mathcal{F}(v) \rrbracket_R)$. Since \mathcal{E}_τ is bijective, we have $\llbracket \mathcal{F}(u) \rrbracket_R = \llbracket \mathcal{F}(v) \rrbracket_R$ —i.e., $u \sim v$. The terms s, t, u, v are in $\mathcal{T}_{\text{ground}}(\Sigma_I)$, allowing us to apply Lemma 5.81 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., $\llbracket \mathcal{F}(s u) \rrbracket_R = \llbracket \mathcal{F}(t v) \rrbracket_R$ —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 $\llbracket \mathcal{F}(s u) \rrbracket_R = \llbracket \mathcal{F}(t u) \rrbracket_R$. By Lemma 5.73, it follows that $\llbracket \mathcal{F}(s) \rrbracket_R = \llbracket \mathcal{F}(t) \rrbracket_R$, 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 $\varphi(\mathcal{E}_\tau(\llbracket \mathcal{F}(u) \rrbracket_R)) = \mathcal{E}_v(\llbracket \mathcal{F}(s u) \rrbracket_R)$ for all u . Hence, $\mathcal{E}_{\tau \rightarrow v}(\llbracket \mathcal{F}(s) \rrbracket_R) = \varphi$, 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 $\mathcal{U}^{\text{IPG}} = \{\mathcal{D}_\tau \mid \tau \text{ ground}\}$. In particular, by Lemma 5.74, this implies that $\mathcal{D}_o = \{0, 1\} \in \mathcal{U}^{\text{IPG}}$ as needed, where 0 is identified with $\llbracket \perp \rrbracket$ and 1 with $\llbracket \top \rrbracket$. Moreover, we define $\mathcal{J}_{\text{ty}}^{\text{IPG}}(\kappa)(\mathcal{D}_{\bar{\tau}}) = \mathcal{D}_{\kappa(\bar{\tau})}$ for all $\kappa \in \Sigma_{\text{ty}}$, completing the type interpretation of $\mathcal{J}_{\text{ty}}^{\text{IPG}} = (\mathcal{U}^{\text{IPG}}, \mathcal{J}_{\text{ty}}^{\text{IPG}})$ and ensuring that $\mathcal{J}_{\text{ty}}^{\text{IPG}}(o) = \mathcal{D}_o = \{0, 1\}$.

We define the interpretation function \mathcal{J}^{IPG} for symbols $f : \prod \bar{a}_m : \tau$ by $\mathcal{J}^{\text{IPG}}(f, \mathcal{D}_{\bar{v}_m}) = \mathcal{E}(\llbracket \mathcal{F}(f(\bar{v}_m)) \rrbracket_R)$.

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

Let $a, b \in \{0, 1\}$, $u_0 = \perp$, and $u_1 = \top$. Then, by Lemma 5.74,

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

(I7) Let $\mathcal{D}_\tau \in \mathcal{U}^{\text{IPG}}$ 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 $\mathcal{E}(\llbracket \mathcal{F}(u) \rrbracket_R) = a'$ and $\mathcal{E}(\llbracket \mathcal{F}(v) \rrbracket_R) = b'$. Then

$$\mathcal{J}^{\text{IPG}}(\approx, \mathcal{D}_\tau)(a', b') = \mathcal{E}(\llbracket \mathcal{F}(\approx(\tau)) \rrbracket_R)(\mathcal{E}(\llbracket \mathcal{F}(u) \rrbracket_R), \mathcal{E}(\llbracket \mathcal{F}(v) \rrbracket_R)) = \mathcal{E}(\llbracket \mathcal{F}(u \approx(\tau) v) \rrbracket_R)$$

which is 1 if $a' = b'$ and 0 otherwise by Lemma 5.74. (I8) Similarly $\mathcal{J}^{\text{IPG}}(\neq, \mathcal{D}_\tau)(a', b') = 0$ if $a' = b'$ and 1 otherwise. This concludes the proof that \mathcal{J}^{IPG} is an interpretation function.

Finally, we need to define the designation function \mathcal{L}^{IPG} , 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 $\mathcal{E}(\llbracket \mathcal{F}(x\theta) \rrbracket_R) = \xi_{\text{te}}(x)$ 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 $\mathcal{L}^{\text{IPG}}(\xi, \lambda t) = \mathcal{E}(\llbracket \mathcal{F}((\lambda t)\theta) \rrbracket_R)$. 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 5.81 it follows that $\llbracket \mathcal{F}((\lambda t)\theta) \rrbracket_R = \llbracket \mathcal{F}((\lambda t)\theta') \rrbracket_R$, which proves that \mathcal{L}^{IPG} is well defined. This concludes the definition of the interpretation $\mathcal{J}^{\text{IPG}} = (\mathcal{U}^{\text{IPG}}, \mathcal{J}_{\text{ty}}^{\text{IPG}}, \mathcal{J}^{\text{IPG}}, \mathcal{L}^{\text{IPG}})$. It remains to show that \mathcal{J}^{IPG} is proper.

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

Lemma 5.82. *Given a ground λ -term $t \in \mathcal{T}_{\text{ground}}^\lambda(\Sigma_1)$, we have*

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

Proof. The proof is adapted from the proof of Lemma 40 in Bentkamp et al. [8]. We proceed by induction on t . If t is of the form $f(\bar{\tau})$, then

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

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

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

If t is a λ -expression, then

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

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

We need to show that the interpretation \mathcal{J}^{IPG} is proper. In the proof, we will need the following lemma, which is very similar to the substitution lemma (Lemma 4.1), but we must prove it here for our particular interpretation \mathcal{J}^{IPG} because we have not shown that \mathcal{J}^{IPG} is proper yet.

Lemma 5.83. *Let ρ be a grounding substitution, t be a λ -term, and ξ be a valuation. Moreover, we define a valuation ξ' by $\xi'_{\text{ty}}(\alpha) = \llbracket \alpha\rho \rrbracket_{\mathcal{J}^{\text{IPG}}}^{\xi_{\text{ty}}}$ for all type variables α and $\xi'_{\text{te}}(x) = \llbracket x\rho \rrbracket_{\mathcal{J}^{\text{IPG}}}^\xi$ for all term variables x . We then have*

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

Proof. The proof is adapted from the proof of Lemma 41 in Bentkamp et al. [8]. We proceed by induction on the structure of τ and t . The proof is identical to that of Lemma 4.1, 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:

$$\begin{aligned}
 \llbracket (\lambda u)\rho \rrbracket_{\mathcal{J}^{\text{IPG}}}^\xi &= \llbracket \lambda u\rho \rrbracket_{\mathcal{J}^{\text{IPG}}}^\xi \\
 &= \mathcal{L}^{\text{IPG}}(\xi, \lambda u) && \text{by the definition of the term denotation} \\
 &= \mathcal{E}(\llbracket \mathcal{F}((\lambda u)\rho\theta\downarrow_\beta) \rrbracket_R) && \text{for some } \theta \text{ by the definition of } \mathcal{L}^{\text{IPG}} \\
 &= \mathcal{E}(\llbracket \mathcal{F}((\lambda u)\rho\downarrow_\beta) \rrbracket_R) && \text{because } (\lambda u)\rho \text{ is ground} \\
 &\stackrel{*}{=} \mathcal{L}^{\text{IPG}}(\xi', \lambda u) && \text{by the definition of } \mathcal{L}^{\text{IPG}} \text{ and Lemma 5.82} \\
 &= \llbracket \lambda u \rrbracket_{\mathcal{J}^{\text{IPG}}}^{\xi'} && \text{by the definition of the term denotation}
 \end{aligned}$$

The step labeled with $*$ is justified as follows: We have $\mathcal{L}^{\text{IPG}}(\xi', \lambda u) = \mathcal{E}(\llbracket \mathcal{F}((\lambda u)\theta'\downarrow_\beta) \rrbracket_R)$ by the definition of \mathcal{L}^{IPG} , 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 5.82, ρ is such a substitution. Hence, $\mathcal{L}^{\text{IPG}}(\xi', \lambda u) = \mathcal{E}(\llbracket \mathcal{F}((\lambda u)\rho\downarrow_\beta) \rrbracket_R)$. \square

Lemma 5.84. *The interpretation \mathcal{J}^{IPG} is proper.*

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

$$\begin{aligned}
 \llbracket \lambda t \rrbracket_{\mathcal{J}^{\text{IPG}}}^{(\xi_{\text{ty}}, \xi_{\text{te}})}(a) &= \mathcal{L}^{\text{IPG}}((\xi_{\text{ty}}, \xi_{\text{te}}), \lambda t)(a) && \text{by the definition of term denotation} \\
 &= \mathcal{E}(\llbracket \mathcal{F}((\lambda t)\theta\downarrow_\beta) \rrbracket_R)(a) && \text{by the definition of } \mathcal{L}^{\text{IPG}} \text{ for some } \theta \\
 &= \mathcal{E}(\llbracket \mathcal{F}(((\lambda t)\theta)s)\downarrow_\beta \rrbracket_R) && \text{such that } \mathcal{E}(\llbracket \mathcal{F}(z\theta) \rrbracket_R) = \xi_{\text{te}}(z) \text{ for} \\
 & && \text{all } z \text{ and } \mathcal{D}_{\alpha\theta} = \xi_{\text{ty}}(\alpha) \text{ for all } \alpha \\
 &= \mathcal{E}(\llbracket \mathcal{F}(t\{0 \mapsto x\}(\theta[x \mapsto s])\downarrow_\beta) \rrbracket_R) && \text{by the definition of } \mathcal{E} \\
 & && \text{where } \mathcal{E}(\llbracket \mathcal{F}(s) \rrbracket_R) = a \\
 &= \llbracket t\{0 \mapsto x\}(\theta[x \mapsto s]) \rrbracket_{\mathcal{J}^{\text{IPG}}} && \text{by } \beta\text{-reduction} \\
 & && \text{where } x \text{ is fresh} \\
 &= \llbracket t\{0 \mapsto x\} \rrbracket_{\mathcal{J}^{\text{IPG}}}^{(\xi_{\text{ty}}, \xi_{\text{te}}[x \mapsto a])} && \text{by Lemma 5.82} \\
 & && \text{by Lemma 5.83}
 \end{aligned}$$

\square

Lemma 5.85. *\mathcal{J}^{IPG} is term-generated; i.e., for all $\mathcal{D} \in \mathcal{U}^{\text{IPG}}$ and all $a \in \mathcal{D}$, there exists a ground type τ such that $\llbracket \tau \rrbracket_{\mathcal{J}^{\text{IPG}}_{\text{ty}}} = \mathcal{D}$ and a ground term t such that $\llbracket t \rrbracket_{\mathcal{J}^{\text{IPG}}} = a$.*

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 $s \in \mathcal{T}_{\text{PF}}$ such that $\llbracket s \rrbracket_R = \mathcal{E}^{-1}(a)$. Let $t = \mathcal{F}^{-1}(s)$. Then, by Lemma 5.82, $\llbracket t \rrbracket_{\mathcal{J}^{\text{IPG}}} = \mathcal{E}(\llbracket s \rrbracket_R) = a$. \square

Lemma 5.86. *Given $C \cdot \theta \in \mathcal{C}_{\text{IPG}}$, we have $\mathcal{J}^{\text{IPG}} \models C \cdot \theta$ if and only if $R \models \mathcal{F}(C \cdot \theta)$.*

Proof. By Lemma 5.82, we have

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

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

Theorem 5.87. *Let $N \subseteq \mathcal{C}_{\text{IPG}}$ be saturated up to redundancy w.r.t. IPGRed_I , and N does not contain a closure of the form $\perp \cdot \theta$ for any θ . Then $\mathcal{I}^{\text{IPG}} \models \text{irred}_R(N)$, where $R = R_{\mathcal{F}(N)}$.*

Proof. By Lemma 5.86, it suffices to show that R is a model of $\text{irred}_R(\mathcal{F}(N))$. We apply Theorem 5.78. Lemma 5.36 shows the condition of saturation up to redundancy. \square

Lemma 5.88. *Let R be a confluent term rewrite system on \mathcal{T}_{PF} oriented by $\succ_{\mathcal{F}}$ whose only Boolean normal forms are \top and \perp . Let $N \subseteq \mathcal{C}_{\text{IPG}}$ such that for every $C \cdot \theta \in N$ and every grounding substitution ρ that coincides with θ on all variables not occurring in C , we have $C \cdot \rho \in N$. Then $R \cup \mathcal{F}(\text{irred}_R(N)) \models_{\text{o}\lambda} \mathcal{F}(N)$.*

Proof. We apply Lemma 5.79. The required condition on $\mathcal{F}(N)$ can be derived from this lemma's condition on N and the fact that \mathcal{F} is a bijection (Lemma 5.4). \square

5.6.3. Partly Substituted Ground Higher-Order Level. In this subsubsection, let \succ be an admissible term order for PGInf (Definition 5.22), and let pgsel be a selection function on \mathcal{C}_{PG} (Definition 5.18).

It is inconvenient to construct a model of N_0 for the PG level because \mathcal{J} 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 PG level only constructs a model of $\text{irred}_R(\mathcal{J}(N)) \subseteq \mathcal{C}_{\text{IPG}}$ instead of $\text{irred}_R(N) \subseteq \mathcal{C}_{\text{PG}}$. 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 5.89. *Let $N \subseteq \mathcal{C}_{\text{PG}}$ be saturated up to redundancy w.r.t. PGRed_I , and N does not contain a closure of the form $\perp \cdot \theta$ for any θ . Then $\mathcal{I}^{\text{IPG}} \models \mathcal{J}(\text{irred}_R(N))$, where $R = R_{\mathcal{F}(\mathcal{J}(N))}$.*

Proof. This follows from Theorem 5.87 and Lemma 5.43. \square

Lemma 5.90. *Let R be a confluent term rewrite system on \mathcal{T}_{PF} oriented by $\succ_{\mathcal{J}\mathcal{F}}$ whose only Boolean normal forms are \top and \perp . Let $N \subseteq \mathcal{C}_{\text{PG}}$ be a clause set without parameters such that for every $C \cdot \theta \in N$ and every grounding substitution ρ that coincides with θ on all variables not occurring in C , we have $C \cdot \rho \in N$. Then $R \cup \mathcal{F}(\mathcal{J}(\text{irred}_R(N))) \models_{\text{o}\lambda} \mathcal{F}(\mathcal{J}(N))$.*

Proof. We apply Lemma 5.88. The required condition on $\mathcal{J}(N)$ can be derived from this lemma's condition on N as follows. We must show that for every $C \cdot \theta \in \mathcal{J}(N)$ and every grounding substitution ρ that coincides with θ on all variables not occurring in C , we have $C \cdot \rho \in \mathcal{J}(N)$. The closure $C \cdot \theta \in \mathcal{J}(N)$ must be of the form $\mathcal{J}(C' \cdot \theta')$ with $C' \cdot \theta' \in N$. Define ρ' as $x\rho' = \mathcal{J}^{-1}(x\rho)$ for all x . By this lemma's condition on N , it follows that $C' \cdot \rho' \in N$, and thus $C \cdot \rho = \mathcal{J}(C' \cdot \rho') \in \mathcal{J}(N)$. Here, it is crucial that N does not contain parameters because only this guarantees that $C = \mathcal{J}_{\theta'}(C') = \mathcal{J}_{\rho'}(C')$. \square

5.6.4. Full Higher-Order Level. In this subsubsection, let \succ be an admissible term order (Definition 3.16), extended to be an admissible term order for $PGInf$ as in Section 5.4.4, and let $hsel$ be a selection function (Definition 3.18).

Definition 5.91. 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 5.92. *The redundancy criteria $HRed_C$ and $HRed_I$ fulfill the following properties, as stated by Waldmann et al. [33]:*

- (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 $concl(\iota) \in N$, 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 closures $C \cdot \theta$, where $C \in \mathcal{C}_H$ as

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

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 confluent term rewrite systems R on \mathcal{T}_F oriented by \succ whose only Boolean normal forms are \top and \perp and all $C \cdot \theta \in \text{irred}_R(\mathcal{G}(C))$, we have

$$R \cup \{\mathcal{F}(E\zeta) \mid E \cdot \zeta \in \text{irred}_R(\mathcal{G}(N)) \text{ and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \models_{o\lambda} \mathcal{F}(C\theta)$$

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 R be a confluent term rewrite system on \mathcal{T}_F oriented by \succ whose only Boolean normal forms are \top and \perp . Let $C \cdot \theta \in \text{irred}_R(\mathcal{FIPG}(C))$. We must show that

$$R \cup \{\mathcal{F}(E\zeta) \mid E \cdot \zeta \in \text{irred}_R(\mathcal{G}(N \setminus N')) \text{ and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \models_{o\lambda} \mathcal{F}(C\theta)$$

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

$$R \cup \{\mathcal{F}(E\zeta) \mid E \cdot \zeta \in \text{irred}_R(\mathcal{G}(N)) \text{ and } E \cdot \zeta \blacktriangleleft C \cdot \theta\} \models_{o\lambda} \mathcal{F}(C\theta)$$

So it suffices to show that

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

Let $E_0 \cdot \zeta_0 \in \text{irred}_R(\mathcal{G}(N))$ 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

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

Our induction hypothesis states:

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

If $E_0 \cdot \zeta_0 \in \text{irred}_R(\mathcal{G}(N \setminus N'))$, the claim $(*)$ is obvious. So we may assume that $E_0 \cdot \zeta_0 \in \text{irred}_R(\mathcal{G}(N'))$. The assumption of (R3) states $N' \subseteq HRed_C(N)$, and thus we have

$$R \cup \{\mathcal{F}(E\zeta) \mid E \cdot \zeta \in \text{irred}_R(\mathcal{G}(N)) \text{ and } E \cdot \zeta \blacktriangleleft E_0 \cdot \zeta_0\} \models_{o\lambda} \mathcal{F}(E_0 \zeta_0)$$

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

(R3) for inferences:

Inspecting this definition of $HRed_I$ (Definition 5.47), we observe that to show that $HRed_I(N) \subseteq HRed_I(N \setminus N')$, it suffices to prove that

$$\begin{aligned} R \cup \{E \in \text{irred}_R(\mathcal{FIPG}(N \setminus N')) \mid E \prec_{\mathcal{F}} \mathcal{F}(C_m \theta_m)\} \\ \models_{o\lambda} \\ R \cup \{E \in \text{irred}_R(\mathcal{FIPG}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(C_m \theta_m)\} \end{aligned}$$

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

$$\begin{aligned} R \cup \{\mathcal{F}(E\zeta) \mid E \cdot \zeta \in \text{irred}_R(\mathcal{G}(N \setminus N')) \text{ and } E\zeta \prec C_m \theta_m\} \\ \models_{o\lambda} \{\mathcal{F}(E\zeta) \mid E \cdot \zeta \in \text{irred}_R(\mathcal{G}(N)) \text{ and } E\zeta \prec C_m \theta_m\} \end{aligned}$$

Let $E_0 \cdot \zeta_0 \in \text{irred}_R(\mathcal{G}(N))$ with $E_0 \zeta_0 \prec C_m \theta_m$. We must show that

$$R \cup \{\mathcal{F}(E\zeta) \mid E \cdot \zeta \in \text{irred}_R(\mathcal{G}(N \setminus N')) \text{ and } E\zeta \prec C_m \theta_m\} \models_{o\lambda} \mathcal{F}(E_0 \zeta_0) \quad (\dagger)$$

If $E_0 \cdot \zeta_0 \in \text{irred}_R(\mathcal{G}(N \setminus N'))$, the claim (\dagger) is obvious. So we may assume that $E_0 \cdot \zeta_0 \in \text{irred}_R(\mathcal{G}(N'))$. 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

$$R \cup \{\mathcal{F}(E\zeta) \mid E \cdot \zeta \in \text{irred}_R(\mathcal{G}(N \setminus N')) \text{ and } E \cdot \zeta \blacktriangleleft E_0 \cdot \zeta_0\} \models_{o\lambda} \mathcal{F}(E_0 \zeta_0)$$

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 $\text{concl}(\iota) \in N$. We must show that $\iota \in HRed_I(N)$. Let $C_1[\![S_1]\!]$, \dots , $C_m[\![S_m]\!]$ be ι 's premises and $C_{m+1}[\![S_{m+1}]\!]$ its conclusion. Let $\theta_1, \dots, \theta_{m+1}$ be a tuple of substitutions for which ι is rooted in $FInf$ (Definition 3.31). Let R be a confluent term rewrite system oriented by $\succ_{\mathcal{F}}$ whose only Boolean normal forms are \top and \perp such that $C_{m+1} \cdot \theta_{m+1}$ is variable-irreducible. According to the definition of $HRed_I$ (Definition 5.47), we must show that

$$R \cup O \models_{o\lambda} \mathcal{F}(C_{m+1} \theta_{m+1})$$

where $O = \text{irred}_R(\mathcal{FIPG}(N))$ if ι is a DIFF inference and $O = \{E \in \text{irred}_R(\mathcal{FIPG}(N)) \mid E \prec_{\mathcal{F}} \mathcal{F}(C_m \theta_m)\}$ if ι is some other inference.

Since $\text{concl}(\iota) \in N$ and $\text{concl}(\iota) = C_{m+1}[\![S_{m+1}]\!]$, we have $C_{m+1}[\![S_{m+1}]\!] \in N$. Thus, by Lemma 5.17, $\mathcal{F}(C_{m+1} \theta_{m+1}) \in \mathcal{FIPG}(N)$. Since $C_{m+1} \cdot \theta_{m+1}$ is variable-irreducible, we have $\mathcal{F}(C_{m+1} \theta_{m+1}) \in \text{irred}_R(\mathcal{FIPG}(N))$. This completes the proof for DIFF inferences because $\mathcal{F}(C_{m+1} \theta_{m+1}) \models_{o\lambda} \mathcal{F}(C_{m+1} \theta_{m+1})$. For the other inferences, it remains to prove that $\mathcal{F}(C_{m+1} \theta_{m+1}) \prec_{\mathcal{F}} \mathcal{F}(C_m \theta_m)$.

By Definition 3.31, $\mathcal{F}(C_m \theta_m)$ 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.

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 $t' = \top$, 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 $t' = \top$. 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)_{PF} and (O4)_{PF}. Thus the premise is larger than the conclusion.

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

For FFALSEELIM, clearly, $C' \vee \perp \approx \top \succ_{\mathcal{F}} C'$.

For FARGCONG, the premise is larger than the conclusion by (O5)_{PF}.

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

Lemma 5.93. *Let R be a confluent term rewrite system on \mathcal{T}_{PF} oriented by $\succ_{\mathcal{F}}$ whose only Boolean normal forms are \top and \perp . Let $N \subseteq \mathcal{C}_G$ be a clause set without parameters such that for every $C \cdot \theta \in N$ and every grounding substitution ρ , we have $C \cdot \rho \in N$. Then $R \cup \mathcal{F}(\mathcal{I}(\mathcal{P}(\text{irred}_R(N)))) \models_{\text{o}\lambda} \mathcal{F}(\mathcal{I}(\mathcal{P}(N)))$.*

Proof. We apply Lemma 5.90. The required condition on $\mathcal{I}(N)$ can be derived from this lemma's condition on N as follows. We must show that for every $C \cdot \theta \in \mathcal{P}(N)$ and every grounding substitution ρ that coincides with θ on all variables not occurring in C , we have $C \cdot \rho \in \mathcal{P}(N)$. The closure $C \cdot \theta \in \mathcal{P}(N)$ must be of the form $C' \mathfrak{p}(\theta') \cdot \mathfrak{q}(\theta')$ with $C' \cdot \theta' \in N$. Define $\rho' = \mathfrak{p}(\theta')\rho$. By this lemma's condition on N , it follows that $C' \cdot \rho' \in N$. By Lemma 5.13, $\mathfrak{p}(\rho') = \mathfrak{p}(\theta')$. We have $y\rho = y\theta = y\mathfrak{q}(\theta')$ for all variables y not occurring in C and in particular for all y not introduced by $\mathfrak{p}(\theta')$. Thus, by Lemma 5.14, $\mathfrak{q}(\rho') = \rho$. So, $C \cdot \rho = \mathcal{P}(C' \cdot \rho') \in \mathcal{P}(N)$. \square

Theorem 5.94. *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, and
3. N_0 does not contain constraints,

we have $\perp[\![S]\!] \in N_i$ for some satisfiable constraints S and some index i .

Proof. By Lemma 9 of Waldmann et al. [33], using Lemma 5.92, the limit $N_\infty = \bigcup_i \bigcap_{j \geq i} N_j$ is saturated up to redundancy w.r.t. $H\text{Inf}$ and $H\text{Red}_1$. By Lemma 5.53, $\mathcal{PG}(N_\infty)$ is saturated up to redundancy w.r.t. $P\text{GInf}$ and $P\text{GRed}_1$.

For a proof by contradiction, assume that for all S and all i , $\perp[\![S]\!] \notin N_i$. Then N_∞ does not contain such a clause $\perp[\![S]\!]$ either, and thus $\mathcal{PG}(N_\infty)$ does not contain a clause of the form $\perp \cdot \theta$ for any θ . By Lemma 5.89, $\mathcal{J}^{\text{IPG}} \models \text{irred}_R(\mathcal{I}(\mathcal{PG}(N_\infty)))$, where $R = R_{\mathcal{F}\mathcal{IPG}(N_\infty)}$.

By Lemma 8 of Waldmann et al. [33], using Lemma 5.92, $N_0 \subseteq N_\infty \cup H\text{Red}_C(N_\infty)$. Thus, $R \cup \text{irred}_R(\mathcal{F}\mathcal{IPG}(N_\infty)) \models_{\text{o}\lambda} \text{irred}_R(\mathcal{F}\mathcal{IPG}(N_0))$. By Lemma 5.93 and conditions 2 and 3 from the present theorem, $R \cup \text{irred}_R(\mathcal{F}\mathcal{IPG}(N_0)) \models_{\text{o}\lambda} \mathcal{F}\mathcal{IPG}(N_0)$ and thus $R \cup \text{irred}_R(\mathcal{F}\mathcal{IPG}(N_\infty)) \models_{\text{o}\lambda} \mathcal{F}\mathcal{IPG}(N_0)$. Since $\mathcal{J}^{\text{IPG}} \models \text{irred}_R(\mathcal{I}(\mathcal{PG}(N_\infty)))$, by Lemma 5.86, it follows that $\mathcal{J}^{\text{IPG}} \models \mathcal{I}(\mathcal{P}(N_0))$.

If we applied each closure's substitution to its clause in the sets $\mathcal{I}(\mathcal{P}(N_0))$ and $\mathcal{I}(\text{Gnd}(N_0))$, the two sets would be identical. So, since $\mathcal{J}^{\text{IPG}} \models \mathcal{I}(\mathcal{P}(N_0))$, we have $\mathcal{J}^{\text{IPG}} \models \mathcal{I}(\text{Gnd}(N_0))$.

Now \mathcal{J}^{IPG} 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}^{IPG} is term-generated by Lemma 5.85, there exists a substitution θ such that $\llbracket \alpha\theta \rrbracket_{\mathcal{J}^{\text{IPG}}} = \xi_{\text{ty}}(\alpha)$ for all type variables α in C and $\llbracket x\theta \rrbracket_{\mathcal{J}^{\text{IPG}}} = \xi_{\text{te}}(x)$ for all term variables x in C . Since C does not contain parameters by condition 2 of this theorem, $C\theta \in \mathcal{J}(\text{Gnd}(N_0))$. Thus we have $\mathcal{J}^{\text{IPG}} \models C\theta$. By Lemma 4.2, it follows that C is true w.r.t. ξ and \mathcal{J}^{IPG} . Since ξ and $C \in N_0$ were arbitrary, we have $\mathcal{J}^{\text{IPG}} \models N_0$. This contradicts condition 1 of the present theorem. \square

Lemma 5.95. *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 2.1).

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

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

we have $\perp \llbracket S \rrbracket \in N_i$ for some satisfiable constraints S and some index i .

Proof. By Theorem 5.94 and Lemma 5.95. \square

6. CONCLUSION

We presented the optimistic λ -superposition calculus. It is inspired by the original λ -superposition calculus of Bentkamp et al. [6], which in turn generalizes the standard superposition calculus by Bachmair and Ganzinger [1]. Our calculus has many advantages over the original λ -superposition calculus, including more efficient handling of unification, functional extensionality, and redundancy. Admittedly, its main disadvantage is its lengthy refutational completeness proof.

We have some ideas on how to extend the calculus further:

- We believe that the inference rules that still require full unification could be adapted to work with partial unification by adding annotations to constrained clauses. The annotations would indicate which variables and which constraints stem from rules with the FLUID- prefix. A modification of the map \mathfrak{p} used in our proof could ensure that these variables do not carry the guarantee of being variable-irreducible that currently

all variables carry. As a result, the proof of Lemma 5.53 would no longer require full unification, but additional FLUIDSUP inferences would be required into the variables marked by the annotations.

- We conjecture that the DIFF axiom is not necessary for refutational completeness although our proof currently requires it. Our proof uses it in Lemma 5.80 to show that the constructed model is a valid higher-order model in the sense that equality of functions implies equality of their values on all arguments. We suspect that one can construct a model with this property using saturation w.r.t. ARGCONG alone, but the model construction must be different from the one used in the present proof.
- One of the most explosive rules of the calculus is FLUIDSUP. Bhayat and Suda [11] propose a modification of inference rules that delays flex-rigid pairs and flex-flex pairs by adding them as negative literals to the conclusion. They suggest that this modification in conjunction with additional inference rules for the unification of flex-rigid pairs could remove the need for FLUIDSUP. We conjecture that one could prove refutational completeness of such a calculus by restructuring Lemma 5.77 to apply the modified inference rules instead of Lemma 5.76 whenever the only terms reducible by R_C correspond to positions below applied variables on level H.
- Similarly, we conjecture that one could remove the EXT rule by following the idea of Bhayat [10] to delay unification of functional terms by adding them as negative literals to the conclusion. If we immediately apply NEGEXT to these additional literals, one can possibly prove refutational completeness by restructuring Lemma 5.77 to apply the modified inference rules instead of Lemma 5.76 whenever the only terms reducible by R_C are functional terms.

Acknowledgment. We thank Ahmed Bhayat, Massin Guerdi, and Martin Desharnais for suggesting textual improvements. We thank Nicolas Peltier and Maria Paola Bonacina who we discussed some early ideas with.

Bentkamp and Blanchette’s research has received funding from the European Research Council (ERC, Matryoshka, 713999 and Nekoka, 101083038) Bentkamp’s research has received funding from a Chinese Academy of Sciences President’s International Fellowship for Postdoctoral Researchers (grant No. 2021PT0015) and from the program Freiraum 2022 of the Stiftung Innovation in der Hochschullehre (ADAM: Anticipating the Digital Age of Mathematics, FRFMM-83/2022). Blanchette’s research has received funding from the Netherlands Organization for Scientific Research (NWO) under the Vidi program (project No. 016.Vidi.189.037, Lean Forward). Hetzenberger’s research has received funding from the European Research Council (ERC, ARTIST, 101002685).

Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

We have used artificial intelligence tools for textual editing.

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] Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic paramodulation and superposition. In Deepak Kapur, editor, *CADE-11*, volume 607 of *LNCS*, pages 462–476. Springer, 1992.
- [4] Alexander Bentkamp, Jasmin Blanchette, and Matthias Hetzenberger. Term orders for optimistic superposition (unpublished manuscript). https://nekoka-project.github.io/pubs/optimistic_orders.pdf.
- [5] 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.
- [6] Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, and Petar Vukmirović. Superposition for higher-order logic. *J. Autom. Reason.*, 67(1):10, 2023.
- [7] 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.
- [8] Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, Petar Vukmirovic, and Uwe Waldmann. Superposition with lambdas. *J. Autom. Reason.*, 65(7):893–940, 2021.
- [9] Christoph Benzmüller, Nik Sultana, Lawrence C. Paulson, and Frank Theiss. The higher-order prover LEO-II. *J. Autom. Reason.*, 55(4):389–404, 2015.
- [10] Ahmed Bhayat. *Automated theorem proving in higher-order logic*. PhD thesis, University of Manchester, 2021.
- [11] Ahmed Bhayat and Martin Suda. A higher-order vampire (short paper). In *IJCAR (1)*, volume 14739 of *LNCS*, pages 75–85. Springer, 2024.
- [12] Frédéric Blanqui, Jean-Pierre Jouannaud, and Albert Rubio. The computability path ordering. *Log. Meth. Comput. Sci.*, 11(4), 2015.
- [13] Arthur Charguéraud. The locally nameless representation. *J. Autom. Reason.*, 49(3):363–408, 2012.
- [14] 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.
- [15] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, 1979.
- [16] Gilles Dowek. Higher-order unification and matching. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume II, pages 1009–1062. Elsevier and MIT Press, 2001.
- [17] Melvin Fitting. *Types, Tableaus, and Gödel’s God*. Kluwer, 2002.
- [18] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge University Press, 1993.
- [19] Gérard P. Huet. A unification algorithm for typed lambda-calculus. *Theor. Comput. Sci.*, 1(1):27–57, 1975.
- [20] 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.
- [21] Dénes König. Über eine Schlussweise aus dem Endlichen ins Unendliche. *Acta Sci. Math. (Szeged)*, 3499/2009(3:2–3):121–130, 1927.
- [22] Dale Miller. Unification under a mixed prefix. *J. Symb. Comput.*, 14(4):321–358, 1992.
- [23] Robert Nieuwenhuis and Albert Rubio. Basic superposition is complete. In Bernd Krieg-Brückner, editor, *ESOP ’92*, volume 582 of *LNCS*, pages 371–389. Springer, 1992.
- [24] Robert Nieuwenhuis and Albert Rubio. Theorem proving with ordering and equality constrained clauses. *J. Symb. Comput.*, 19(4):321–351, 1995.
- [25] 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.
- [26] 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.
- [27] Stephan Schulz. E - a brainiac theorem prover. *AI Commun.*, 15(2-3):111–126, 2002.

- [28] Geoff Sutcliffe. The 12th IJCAR automated theorem proving system competition—CASC-J12. *The European Journal on Artificial Intelligence*, 38(1):3–20, 2025.
- [29] Geoff Sutcliffe and Martin Desharnais. The CADE-29 automated theorem proving system competition—CASC-29. *AI Commun.*, 37(4):485–503, 2024.
- [30] Petar Vukmirović, Alexander Bentkamp, Jasmin Blanchette, Simon Cruanes, Visa Nummelin, and Sophie Tourret. Making higher-order superposition work. *J. Autom. Reason.*, 66(4):541–564, 2022.
- [31] Petar Vukmirović, Alexander Bentkamp, and Visa Nummelin. Efficient full higher-order unification. *Log. Methods Comput. Sci.*, 17(4), 2021.
- [32] Petar Vukmirović, Jasmin Blanchette, and Stephan Schulz. Extending a high-performance prover to higher-order logic. In *TACAS 2023*, volume 13994 of *LNCS*, pages 111–129. Springer, 2023.
- [33] Uwe Waldmann, Sophie Tourret, Simon Robillard, and Jasmin Blanchette. A comprehensive framework for saturation theorem proving. *J. Autom. Reason.*, 66(4):499–539, 2022.