

A Complete Superposition Calculus for Higher-Order Logic

Ahmed Bhayat and Giles Reger

University of Manchester, Manchester, UK

Abstract. We present a refutationally complete superposition calculus for a version of higher-order logic (HOL) based on the combinatory calculus. We also introduce a novel method of dealing with extensionality. The calculus was implemented in the Vampire theorem prover and we test its performance against other leading higher-order provers. The results suggest that the method is competitive.

1 Introduction

First-order superposition provers are often used to reason about problems in extensional higher-order logic [4][6]. Commonly, this is achieved by translating the higher-order problem to first-order logic using combinators. Such a strategy is sub-optimal because translations generally sacrifice completeness and at times even soundness. In this paper, we provide a modification of first-order superposition that is sound and complete for a combinatory version of higher-order logic. Moreover, it is *graceful* in the sense of [2].

The work is complementary to that of Bentkamp et al. in [2]. However, our approach appears to offer two clear differences. Firstly, as our calculus is based on the combinatory logic and first-order unification, it is far closer to standard first-order superposition. Therefore, it should be easier to implement in state of the art first-order provers. Secondly, the $>_{\text{ski}}$ ordering that we propose to parameterise our calculus with can compare more terms than can be compared by the ordering presented in [2]. On the other hand, we suspect that for problems requiring complex unifiers, our approach will not be competitive with lambda superposition.

Developing a complete and efficient superposition calculus for a combinatory version of higher-order logic poses some difficulties. When working with a monomorphic logic it is impossible to select a finite set of typed combinator axioms that can guarantee completeness for a particular problem [3]. Secondly, using existing orderings, combinator axioms can superpose among themselves, leading to a huge number of consequences of the axioms. If the problem is first-order, these consequences can never interact with non-combinator clauses and are therefore useless.

We deal with both issues in the current work. To circumvent the first issue, we base our calculus on a polymorphic rather than monomorphic first-order logic. The second issue can be dealt with by an ordering that orients combinator axioms left-to-right. Consider the **S**-combinator axiom $\mathbf{S} x y z \approx x z (y z)$. Assume that there exists a simplification ordering \succ such that $\mathbf{S} x y z \succ x z (y z)$ exists. Then, since superposition is only carried out on the larger side of literals and not at variables, there can be no inferences between the **S**-axiom and any other combinator axiom. Indeed, in this case

the axioms can be removed from the clause set altogether and replaced by an inference rule (Section 9).

Unfortunately no simplification ordering is known that is capable of orienting all axioms for a complete set of combinators.¹ The authors suspect that no such simplification ordering can exist. Consider a KBO-like ordering. Since, the variable x appears twice on the right-hand side of the **S**-axiom and only once on the left-hand side, the ordering would not be able to orient it. The same is the case for any other combinator which duplicates its arguments.

In other work REF, we have developed an ordering that enjoys most of the properties of a simplification ordering, but lacks full compatibility with contexts. In particular, the ordering is not compatible with what we call *unstable* contexts. We propose using such an ordering to parameterise the superposition calculus. In the standard proof of the completeness of superposition, compatibility with contexts is used to rule out the need for superposition at or beneath variables. As the ordering doesn't enjoy full compatibility with contexts, limited superposition at and below variables needs to be carried out. This is dealt with by the addition of an extra inference rule to the standard rules of superposition which we call SUBVARSUP (Section 3).

By turning combinator axioms into rewrite rules, the calculus represents a folding of higher-order unification into the superposition calculus itself. Whilst not as goal-directed as a dedicated higher-order unification algorithm, it is still far more goal-directed than using combinators in superposition provers along with standard orderings. Consider the conjecture $\exists z. \forall xy. zxy \approx fyx$. Bentkamp et al. ran an experiment and found that the E prover [5] running on this conjecture supplemented with the **S**- and **K**-combinator axioms had to perform 3756 inferences in order to find a refutation [2]. Our calculus reduces this number to 427 inferences, despite the fact that our calculus is complete for HOL whilst E presumably was not.

say what HOL
stands for

With the addition of rewrite rules for **C**-, **B**- and **I**-combinators, the number of inferences required reduces to 18. We consider likely that for problems requiring 'simple' unifiers, folding unification into superposition will be competitive with higher-order unification whilst providing the advantages that data-structures and algorithms developed for first-order superposition can be re-used unchanged. The results of the empirical evaluation of our method can be found in Section 10.

2 The Logic

The logic we use is polymorphic applicative first-order logic otherwise known as λ -free higher-order logic.

Syntax Let \mathcal{V}_{ty} be a set of type variables and Σ_{ty} be a set of type constructors with fixed arities. It is assumed that a binary type constructor \rightarrow is present in Σ_{ty} which is written infix. The set of types is defined:

Polymorphic Types $\tau ::= \kappa(\overline{\tau_n}) \mid \alpha \mid \tau \rightarrow \tau$ where $\alpha \in \mathcal{V}_{ty}$ and $\kappa \in \Sigma_{ty}$

¹ A complete set of combinators is a set of combinators whose members can be composed to form a term extensionally equivalent to any given λ -term.

The notation $\overline{t_n}$ is used to denote a tuple or list of types or terms depending on the context. A type declaration is of the form $\Pi \overline{\alpha_n} . \sigma$ where σ is a type and all type variables in σ appear in $\overline{\alpha}$. Let Σ be a set of typed function symbols and \mathcal{V} a set of variables with associated types. It is assumed that Σ contains the following function symbols, known as *basic combinators*:

$$\begin{aligned} \mathbf{S} &: \Pi \alpha \tau \gamma. (\alpha \rightarrow \tau \rightarrow \gamma) \rightarrow (\alpha \rightarrow \tau) \rightarrow \alpha \rightarrow \gamma \\ \mathbf{C} &: \Pi \alpha \tau \gamma. (\alpha \rightarrow \tau \rightarrow \gamma) \rightarrow \tau \rightarrow \alpha \rightarrow \gamma \\ \mathbf{B} &: \Pi \alpha \tau \gamma. (\alpha \rightarrow \gamma) \rightarrow (\tau \rightarrow \alpha) \rightarrow \tau \rightarrow \gamma \\ \mathbf{K} &: \Pi \alpha \gamma. \alpha \rightarrow \gamma \rightarrow \alpha \\ \mathbf{I} &: \Pi \alpha. \alpha \rightarrow \alpha \end{aligned}$$

How to introduce combinators nicely?

The set of terms over Σ and \mathcal{V} is defined below. In what follows, type subscripts are generally omitted.

Terms $\mathcal{T} ::= x \mid f(\overline{t_n}) \mid t_{\tau' \rightarrow \tau} t'_{\tau'}$ where $f : \Pi \overline{\alpha_n} . \sigma \in \Sigma, x \in \mathcal{V}$ and $t, t' \in \mathcal{T}$

The type of the term $f(\overline{t_n})$ is $\sigma\{\overline{\alpha_n} \rightarrow \overline{t_n}\}$. Following [?], terms of the form $t_1 t_2$ are called applications. Non-application terms are called heads. A term can uniquely be decomposed into a head and n arguments. Let $t = \zeta \overline{t'_n}$. Then $head(t) = \zeta$ where ζ could be a variable or constant applied to possibly zero type arguments. The symbol $\tilde{\mathbf{C}}$ denotes an arbitrary combinator, whilst \mathbf{C} denotes a member of $\{\mathbf{S}, \mathbf{C}, \mathbf{B}\}$. These symbols are only used when the combinator is assumed to have a full complement of arguments. Thus, in $\mathbf{C} \overline{t_n}$, $n \geq 3$ is assumed. The symbols $x, y, z \dots$ are reserved for variables, $c, d, f \dots$ for non-combinator constants and ζ, ξ range over arbitrary symbols and, by an abuse of notation, at times even terms.

Positions over terms: for a term t , if $t \in \mathcal{V}$ or $t = f(\overline{t_n})$, then $pos(t) = \{\epsilon\}$ (type arguments have no position). If $t = t_1 t_2$ then $pos(t) = \{\epsilon\} \cup \{i.p \mid 1 \leq i \leq 2, p \in pos(t_i)\}$. Subterms at positions of the form $p.1$ are called *prefix* subterms. We define *first-order* subterms inductively as follows. For any term t , t is a first-order subterm of itself. If $t = \zeta \overline{t_n}$, where ζ is not a fully applied combinator, then the first-order subterms of each t_i are also first-order subterms of t . the notation $s\langle u \rangle$ is to be read as u is a first-order subterm of s . Note that this definition is subtly different to that in REF since subterms underneath a fully applied combinator are not considered to be first-order.

Stable subterms: let $FNP(t, p)$ be a partial function that takes a term t , a position p and returns the longest prefix p' of p such that $head(t|_{p'})$ is not a partially applied combinator if such a position exists. For a position $p \in pos(t)$, p is a stable position in t if p is not a prefix position and $FNP(t, p)$ is not defined or $head(t|_{FNP(t, p)})$ is not a variable or combinator. A *stable subterm* is a subterm occurring at a stable position. For example, the subterm a is not stable in $f(\mathbf{S} a b c)$, $\mathbf{S}(\mathbf{S} a) b c$ (in both cases, $head(t|_{FNP(t, p)}) = \mathbf{S}$) and $a c$ (a is in a prefix position), but is in $g a b$ and $f(\mathbf{S} a) b$. A subterm that is not stable is known as an *unstable* subterm.

The notation $t[u]$ denotes an arbitrary subterm u of t that may be unstable. The notation $t[u_1, \dots, u_n]_n$, at times given as $t[\overline{u}]_n$ denotes that the term t contains n non-overlapping subterms u_1 to u_n . By $u[]_n$, we refer to a context with n non-overlapping holes.

Weak reduction: each combinator is defined by its characteristic equation; $\mathbf{S} x y z = x z (y z)$, $\mathbf{C} x y z = x z y$, $\mathbf{B} x y z = x (y z)$, $\mathbf{K} x y = x$ and $\mathbf{I} x = x$. A term t *weak-reduces* to a term t' in one step (denoted $t \rightarrow_w t'$) if $t = u[s]_p$ and there exists a combinator axiom $l = r$ and substitution σ such that $l\sigma = s$ and $t' = u[r\sigma]_p$. The term $l\sigma$ in t is called a *weak redex* or just redex. By \rightarrow_w^* , the reflexive transitive closure of \rightarrow_w is denoted. Weak-reduction is terminating and confluent as proved in [?]. By $(t) \downarrow^w$, we denote the term formed from t by contracting its leftmost redex.

An equation $s \approx t$ is an unordered pair of terms and a literal is an equation or the negation of an equation represented $s \not\approx t$. A clause is a multiset of literals represented $L_1 \vee \dots \vee L_n$ where each L_i is a literal.

Semantics We follow Bentkamp et al. [1] closely in specifying the semantics. An interpretation is a triple $(\mathcal{U}, \mathcal{E}, \mathcal{J})$ where \mathcal{U} is a ground-type indexed family of non-empty sets called *universes* and \mathcal{E} is a family of functions $\mathcal{E}_{\tau, v} : \mathcal{U}_{\tau \rightarrow v} \rightarrow (\mathcal{U}_{\tau} \rightarrow \mathcal{U}_v)$. A *type valuation* ξ is a substitution that maps type variables to ground types and whose domain is the set of all type variables. A type valuation ξ is extended to a *valuation* by setting $\xi(x_{\tau})$ to be a member of $\mathcal{U}_{(\tau\xi)}$. An interpretation function \mathcal{J} maps a function symbol $f : \prod \overline{\alpha_n} \cdot \sigma$ and a tuple of ground types $\overline{\tau_n}$ to a member of $\mathcal{U}_{(\sigma\{\alpha_i \rightarrow \tau_i\})}$. An interpretation is *extensional* if $\mathcal{E}_{\tau, v}$ is injective for all τ, v and is *standard* if $\mathcal{E}_{\tau, v}$ is bijective for all τ, v .

For an interpretation $\mathcal{I} = (\mathcal{U}, \mathcal{E}, \mathcal{J})$ and a valuation ξ , a term is denoted as follows: $\llbracket x \rrbracket_{\mathcal{I}}^{\xi} = \xi(x)$, $\llbracket f(\overline{\tau}) \rrbracket_{\mathcal{I}}^{\xi} = \mathcal{J}(f, \llbracket \overline{\tau} \rrbracket_{\mathcal{I}}^{\xi})$ and $\llbracket st \rrbracket_{\mathcal{I}}^{\xi} = \mathcal{E}(\llbracket s \rrbracket_{\mathcal{I}}^{\xi})(\llbracket t \rrbracket_{\mathcal{I}}^{\xi})$. An equation $s \approx t$ is true in an interpretation \mathcal{I} with valuation function ξ if $\llbracket s \rrbracket_{\mathcal{I}}^{\xi}$ and $\llbracket t \rrbracket_{\mathcal{I}}^{\xi}$ are the same object and is false otherwise. A disequation $s \not\approx t$ is true if $s \approx t$ is false. A clause is true if one of its literals is true and a clause set is true if every clause in the set is true. An interpretation \mathcal{I} models a clause set N , written $\mathcal{I} \models N$, if N is true in \mathcal{I} for all valuation functions ξ .

3 The Calculus

The calculus is closely based on Bentkamp et al.'s intensional non-purifying calculus [1]. The extensionality axioms can be added if extensionality is required. The axiom of choice is included as we are working with Henkin semantics with choice. The main difference between our calculus and that of [1] is that superposition inferences are not allowed beneath fully applied combinators and an extra inference rule is added to deal with superposition beneath variables.

Term Ordering We also demand that the calculus is parameterised by a partial ordering that is well-founded, total on ground terms, stable under substitutions and has the subterm property \succ which orients all instances of combinator axioms left-to-right. It is an open problem whether a simplification ordering enjoying this last property exists, but it appears unlikely. However, for completeness, compatibility with stable contexts suffices. The $>_{\text{ski}}$ ordering introduced in REF orients all instances of combinator axioms left-to-right and is compatible with stable contexts. It is *not* compatible with arbitrary contexts. For terms t_1 and t_2 such that $t_1 >_{\text{ski}} t_2$, it is not necessarily the case that

$t_1 u >_{\text{ski}} t_2 u$ or that $\mathbf{S} t_1 a b >_{\text{ski}} \mathbf{S} t_2 a b$. We show that by not superposing underneath fully applied combinators and carrying out some restricted superposition beneath variables, this lack of compatibility with arbitrary contexts can be circumvented and does not lead to a loss of completeness. In a number of places in the completeness proof we assume the following conditions on the ordering (satisfied by the $>_{\text{ski}}$ ordering). It may be possible to relax the conditions at the expense of an increased number of inferences.

P1 For terms t, t' such that $t \rightarrow_w t'$, then $t \succ t'$

P2 For terms t, t' such that $t \succ t'$ and $\text{head}(t')$ is first-order, $u[t] \succ u[t']$

prove this in ordering paper?

The ordering \succ is extended to literals and clauses using the multiset extension as explained in REF.

Inference Rules The calculus is further parametrised by a selection function that maps clauses to a subset of its negative literals. Due to the requirements of the completeness proof, if a term $t = x \overline{s_{n>0}}$ is the maximal term in a clause C , then a literal containing x as a first-order subterm may not be selected. A literal L is called *eligible* if it is selected or there are no selected literals in the clause and it is maximal. In the latter case, it is *strictly eligible* if it is strictly maximal. A variable x has a *bad* occurrence in a clause C if it occurs in C at an unstable position. Occurrences of x in C at stable positions are *good*. A clause C is an *extended combinator axiom* if it is either a combinator axiom or the result of an ARGCONG inference on a combinator axiom.

Conventions: Often a clause is written with a single distinguished literal such as $C' \vee t \approx t'$. In this case:

1. The distinguished literal is always eligible.
2. The name of the clause is assumed to be the name of the remainder without the dash.
3. If the clause is involved in an inference, the distinguished literal is the literal that takes part.

Positive and negative superposition:

$$\frac{D' \vee t \approx t' \quad C' \vee [\neg]s\langle u \rangle \approx s'}{(C' \vee D' \vee [\neg]s\langle t' \rangle \approx s')\sigma} \text{ SUP}$$

with the following side conditions:

1. The variable condition holds
2. C is not an extended combinator axiom;
3. $\sigma = \text{mgu}(t, u)$;
4. $t\sigma \not\approx t'\sigma$;
5. $s\langle u \rangle\sigma \not\approx s'\sigma$;
6. $C\sigma \not\approx D\sigma$ or D is an extended combinator axiom;
7. $(t \approx t')\sigma$ is strictly eligible in $D\sigma$;
8. $([\neg]s\langle u \rangle \approx s')\sigma$ is eligible in $C\sigma$, and strictly eligible if it is positive.

Definition 1. Let $l = \tilde{\mathcal{C}} \overline{x_n} \approx r$ be an extended combinator axiom. A term $v \overline{u_m}$ is compatible with $l \approx r$ if $\tilde{\mathcal{C}} = \mathbf{I}$ and $m = n$ or if $\tilde{\mathcal{C}} = \mathbf{K}$ and $m \geq n - 1$ or if $\tilde{\mathcal{C}} \in \{\mathbf{B}, \mathbf{C}, \mathbf{S}\}$ and $m \geq n - 2$.

Variable condition: $u \notin \mathcal{V}$. If $u = x \overline{s_n}$ and D is a combinator axiom, then D and u must be compatible.

Because the term ordering \succ is not compatible with unstable contexts, there are instances when superposition beneath variables must be carried out. The SUBVARSUP rule deals with this.

$$\frac{D' \vee t \approx t' \quad C' \vee [\neg]s\langle y \overline{u_n} \rangle \approx s'}{(C' \vee D' \vee [\neg]s\langle zt' \overline{u_n} \rangle \approx s')\sigma} \text{ SUBVARSUP}$$

condition 6 can be tightened

with the following side conditions in addition to conditions 3 - 8 of SUP:

1. y has another occurrence bad in C ;
2. z is a fresh variable;
3. $\sigma = \{y \rightarrow zt'\}$;
4. t' has a variable or combinator head;
5. $n \leq 1$;
6. D is not an extended combinator axiom.

The EQRES and EQFACT inferences:

$$\frac{C' \vee s \not\approx s'}{C'\sigma} \text{ EQRES} \qquad \frac{C' \vee u' \approx v' \vee u \approx v}{(C' \vee v \not\approx v' \vee u \approx v')\sigma} \text{ EQFACT}$$

For EQRES: $\sigma = mgu(s, s')$ and $(s \not\approx s')\sigma$ is eligible in the premise. For EQFACT: $\sigma = mgu(u, u')$, $u'\sigma \not\approx v'\sigma$, $u\sigma \not\approx v\sigma$, and $(u \approx v)\sigma$ is eligible in the premise.

$$\frac{C' \vee s \approx s'}{C'\sigma \vee (s\sigma)x \approx (s'\sigma)x} \text{ ARGCONG}$$

The literal $s\sigma \approx s'\sigma$ must be eligible in $C\sigma$ and x is a variable that is fresh for C . The unifier σ is the most general type unifier that ensures the well-typedness of the conclusion.

Along with the above inferences, we assume the presence of the (Skolemised) axiom of choice:

$$\neg p(\text{sk}\langle \alpha \rangle p) \vee p(\varepsilon\langle \alpha \rangle p)$$

Where the function symbols sk and ε are of type $\Pi\alpha.(\alpha \rightarrow o) \rightarrow \alpha$.

3.1 Extensionality

The calculus can be both intensional or extensional. For the calculus to be extensional, we provide two possibilities. The first is to add a polymorphic extensionality axiom. Let diff be a polymorphic symbol of type $\Pi\tau_1, \tau_2.(\tau_1 \rightarrow \tau_2) \rightarrow (\tau_1 \rightarrow \tau_2) \rightarrow \tau_1$. The extensionality axiom can be given as:

$$x(\text{diff}\langle \tau_1, \tau_2 \rangle(x, y)) \not\approx y(\text{diff}\langle \tau_1, \tau_2 \rangle(x, y)) \quad \vee \quad x \approx y$$

4 Examples

We provide some examples of how the calculus works. Some of the examples utilised come from Bentkamp et. al.'s paper [2] in order to allow a comparison of the two methods. In all examples, it is assumed that the clause set has been enriched with the combinator axioms.

Example 1. Consider the unsatisfiable clause:

$$x \ a \ b \not\approx x \ b \ a$$

Superposing onto the left-hand side with the extended **K** axiom $\mathbf{K} \ x_1 \ x_2 \ x_3 \approx x_1 \ x_3$ results in the clause $x_1 \ b \not\approx x_1 \ a$. Superposing onto the left-hand side of this clause, this time with the standard **K** axiom adds the clause $x \not\approx x$ from which \perp is derived by an EQRES inference.

Example 2. Consider the unsatisfiable clause set where $f \ a \succ c$:

$$f \ a \approx c \quad h \ (y \ b)(y \ a) \not\approx h \ (g(f \ b))(g \ c)$$

A SUP inference between the **B** axiom and the subterm $y \ b$ of the second clause adds the clause $h(x_1(x_2 \ b))(x_1(x_2 \ a)) \not\approx h(g(f \ b))(g \ c)$ to the set. By superposing onto the subterm $x_2 \ a$ of this clause with the equation $f \ a \approx c$, we derive the clause $h(x_1(f \ b))(x_1 \ c) \not\approx h(g(f \ b))(g \ c)$ from which \perp can be derived by an EQRES inference.

Example 3. Consider the unsatisfiable clause set where $f \ a \succ c$. This example is the combinatory equivalent of Bentkamp et. al.'s Example 6.

$$f \ a \approx c \quad h \ (y \ (\mathbf{B} \ g \ f) \ a) \ y \not\approx h \ (g \ c) \ \mathbf{I}$$

A SUP inference between the extended **I** axiom $\mathbf{I} \ x_1 \ x_2 \approx x_1 \ x_2$ and the subterm $y \ (\mathbf{B} \ g \ f) \ a$ of the second clause adds the clause $h(\mathbf{B} \ g \ f \ a) \ \mathbf{I} \not\approx h(g \ c) \ \mathbf{I}$ to the set. Superposing onto the subterm $\mathbf{B} \ g \ f \ a$ of this clause with the **B** axiom results in the clause $h(g \ (f \ a)) \ \mathbf{I} \not\approx h(g \ c) \ \mathbf{I}$. Superposition onto the subterm $f \ a$ with the first clause of the original set gives $h(g \ c)) \ \mathbf{I} \not\approx h(g \ c) \ \mathbf{I}$ from which \perp can be derived via EQRES.

Note that in Examples 2 and 3, no use is made of SUBVARSUP even though the analogous FLUIDSUP rule is required in Bentkamp et. al.'s calculus. We have been unable to develop an example that requires the SUBVARSUP rule.

5 Redundancy Criteria

We define a weaker logic by an encoding $\lfloor \cdot \rfloor$ of ground terms into first-order terms with $\lceil \cdot \rceil$ as its inverse. The encoding works by indexing each symbol with its type arguments and argument number. For example, $\lfloor f \rfloor = f_0$, $\lfloor f(\bar{\tau})a \rfloor = f_1^{\bar{\tau}}(a_0)$. Fully applied combinators are translated to constants such that syntactically identical terms are translated to the same constant. For example, $\lfloor \mathbf{S} \ t_1 \ t_2 \ t_3 \rfloor = s_0$. The weaker logic is known as the *floor logic* whilst the original logic is called the *ceiling logic*.

An inference is the ground instance of an inference I if it is equivalent to I after the application of some grounding substitution θ to the premise(s) and conclusion of I and the result is still an inference.

A ground ceiling clause C is redundant with respect to a set of ground ceiling clauses N if $\lfloor C \rfloor$ is entailed by clauses in $\lfloor N \rfloor$ smaller than itself. An arbitrary ceiling clause C is redundant to a set of ceiling clauses N if all its ground instances are redundant with respect to $\mathcal{G}_\Sigma(N)$, the set of all ground instances of clauses in N . $Red(N)$ is the set of all clauses redundant with respect to N .

For ground inferences other than ARGCONG, an inference with right premise C and conclusion E is redundant with respect to a set of clauses N if $\lfloor E \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C \rfloor$. A non-ground inference is redundant if all its ground instances are redundant.

An ARGCONG inference from a combinator axiom is redundant with respect to a set of clauses N if its conclusion is in N . For any other ARGCONG inference, it is redundant with respect N if its premise is redundant with respect to N , or its conclusion is in N or redundant with respect to N . A set N is saturated upto redundancy if every inference with premises in N is redundant with respect to N .

6 Refutational Completeness

The completeness proof is presented in the style of Bentkamp et al. [1] who in turn follow the style of Waldmann [7]. Let N be a set of clauses saturated upto redundancy by the above set of inferences. Let $\mathcal{G}_\Sigma(N)$ be the set of all ground instances of N . By the fact that N is saturated upto redundancy and by the definition of inference redundancy for ARGCONG inferences on combinator axioms, it follows that the following extended combinator axioms are in N for all $n \in \mathbb{N}$.

$$\begin{aligned} \mathbf{I} \, x \, \overline{x'_n} &= x \, \overline{x'_n} \\ \mathbf{K} \, x \, y \, \overline{x'_n} &= x \, \overline{x'_n} \\ \mathbf{B} \, x \, y \, z \, \overline{x'_n} &= x \, (y \, z) \, \overline{x'_n} \\ \mathbf{C} \, x \, y \, z \, \overline{x'_n} &= x \, z \, y \, \overline{x'_n} \\ \mathbf{S} \, x \, y \, z \, \overline{x'_n} &= x \, z \, (y \, z) \, \overline{x'_n} \end{aligned}$$

Thus, there exists an infinite set $ECA \subseteq N$ consisting of all extended combinator axioms. The set of all ground instances of clauses in ECA is denoted by $\mathcal{G}_\Sigma(ECA)$ and $\mathcal{G}_\Sigma(ECA) \subseteq \mathcal{G}_\Sigma(N)$. As per [1] we build a model for $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ which can then be lifted to a model of $\mathcal{G}_\Sigma(N)$ and N . We define a term-rewriting system R_∞ and use it to construct an interpretation of $\lfloor \mathcal{G}_\Sigma(N) \rfloor$. We then use induction to prove that this interpretation is a model of $\lfloor \mathcal{G}_\Sigma(N) \rfloor$.

Candidate Interpretation We define the set of rewrite rules R_{ECA} as $\{l \rightarrow r \mid l \approx r \in \lfloor \mathcal{G}_\Sigma(ECA) \rfloor \text{ and } l \succ r\}$. By the condition that the term order orient all instances of

combinator axioms left-to-right, we have that R_{ECA} is the set of all combinator axioms turned into left-to-right rewrite rules.

Lemma 1. *Let $R_{ECA} = R'_{ECA} \cup \{l \rightarrow r\}$. Then, l is not reducible by any rule in R'_{ECA} .*

Proof. Let $l = \lfloor \tilde{\mathcal{C}} \overline{t_n} \rfloor$ and $l' = \lfloor \tilde{\mathcal{C}} \overline{t'_n} \rfloor$ be the left hand side of two difference members of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$. By the definition of the floor translation $\lfloor \cdot \rfloor$ we have that $\lfloor \tilde{\mathcal{C}} \overline{t_n} \rfloor$ and $\lfloor \tilde{\mathcal{C}} \overline{t'_n} \rfloor$ are both constants and $\lfloor \tilde{\mathcal{C}} \overline{t_n} \rfloor \neq \lfloor \tilde{\mathcal{C}} \overline{t'_n} \rfloor$ proving the lemma.

For every clause $C \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$, induction on \succ is used to defined sets of rewrite rules E_C and R_C . Assume that E_D has been defined for all clauses $D \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$ such that $D \prec C$. Then R_C is defined as $R_{ECA} \cup (\bigcup_{D \prec C} E_D)$. The set E_C contains the rewrite rule $s \rightarrow t$ if the following conditions are met. Otherwise $E_C = \emptyset$.

- (a) $C = C' \vee s \approx t$
- (b) $s \approx t$ is strictly maximal in C
- (c) $s \succ t$
- (d) C is false in R_C
- (e) C' is false in $R_C \cup \{s \rightarrow t\}$
- (f) s is not reducible by any rule in R_C

In this case C is called *productive*. Finally, R_∞ is defined as $R_{ECA} \cup (\bigcup_{C \in \lfloor \mathcal{G}_\Sigma(N) \rfloor} E_C)$. Note that due to the definition of R_C and condition (d) an extended combinator axiom is never productive.

Lemma 2. *The rewrite systems R_C and R_∞ are confluent and terminating.*

Proof. Condition (c) ensures that for every rule $s \rightarrow t$ in R_C or R_∞ we have $s \succ t$. By the well-foundedness of \succ we have that R_C and R_∞ must be terminating.

By Lemma 1 there are no critical pairs between rules in R_{ECA} . Using this fact and condition (f), we have that there are no critical pairs between rules in R_C and R_∞ . Absence of critical pairs implies local confluence. Local confluence and termination implies confluence.

Lemma 3. *If a clause D is true in R_D then it is true in R_C for all $C \succ D$ and in R_∞ .*

Proof. As per Waldmann's proof [7]

Lemma 4. *If a clause $D = D' \vee s \approx t$ is productive then D' is false and $s \approx t$ is true in R_C and R_∞ for all $C \succ D$.*

Proof. As per Waldmann's proof [7]

Lemma 5. *Every member of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ is true in R_C for all $C \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$.*

Proof. By construction of R_{ECA} , all members of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ are true in R_{ECA} . Thus, by definition of R_C all members of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ are true in R_C .

Lemma 6 (Lifting non-SUP inferences). *Let $C\theta \in \mathcal{G}_\Sigma(N)$, such that the selected literals in $C\theta$ match those in C . Then, every EQRES, EQFACT and ground instance of an ARGCONG inference from $C\theta$ is the ground instance of a corresponding inference from C .*

Proof. The proof is identical to that in [1] ignoring purification.

Definition 2. *A ground SUP inference between clauses $D\theta = D'\theta \vee t\theta \approx t'\theta$ and $C\theta$ is liftable in three cases:*

1. *The superposed subterm in $C\theta$ is not at or below a variable in C and the variable conditions holds between C and D*
2. *The superposed subterm in $C\theta$ is at a variable y in C which has another occurrence bad in C , $\text{head}(t'\theta)$ is a combinator, and D is not a combinator axiom.*
3. *The superposed subterm in $C\theta$ is below a variable y in C which has another occurrence bad in C , $\text{head}(y\theta)$ and $\text{head}(t'\theta)$ are combinators and D is not a combinator axiom.*

Next, we link liftable SUP inference to non-ground SUP or SUBVARSUP inferences. One of the tricky aspects of the proof is to show that if the inference occurs beneath a variable, the z variable of the SUBVARSUP inference can be used to create a context that weak reduces to an arbitrary context surrounding a term t . To this end, the function $\langle \cdot \rangle$ from λ -terms to combinatory terms is defined:

$$\begin{aligned} \langle \lambda x.x \rangle &= \mathbf{I} \\ \langle \lambda x.t \rangle &= \mathbf{K} \langle t \rangle, \quad x \text{ doesn't occur in } t \\ \langle \lambda x.tt' \rangle &= \mathbf{B} \langle t \rangle \langle \lambda x.t' \rangle, \quad x \text{ only occurs free in } t' \\ \langle \lambda x.tt' \rangle &= \mathbf{C} \langle \lambda x.t \rangle \langle t' \rangle, \quad x \text{ only occurs free in } t \\ \langle \lambda x.tt' \rangle &= \mathbf{S} \langle \lambda x.t \rangle \langle \lambda x.t' \rangle, \quad x \text{ occurs free in } t \text{ and } t' \end{aligned}$$

The function enjoys the easy to prove property that for λ -free terms t_1 and t_2 , $\langle \lambda x.t_1 \rangle t_2 \rightarrow_w t_1 \{x \rightarrow t_2\}$.

Lemma 7 (Lifting SUP inferences). *Let $D\theta$ and $C\theta$ be members of $\mathcal{G}_\Sigma(N)$ where θ is a substitution and the selected literals of $D, C \in N$ correspond to those of $D\theta$ and $C\theta$. Then for every liftable SUP inference between $C\theta$ and $D\theta$, there exists a ground substitution θ' such that $C\theta = C\theta'$, $D\theta = D\theta'$ and the conclusion of the inference between $C\theta$ and $D\theta$ is combinatory congruent to the conclusion of the θ' instance of a SUP or SUBVARSUP inference between C and D .*

Prove that $C\theta' = C\theta$.

Proof. We assume that $D = D' \vee t \approx t'$ and that $C = C' \vee [\neg]s \approx s'$ and that the inference has the form:

$$\frac{D'\theta \vee t\theta \approx t'\theta \quad C'\theta \vee [\neg]s\theta \langle t\theta \rangle|_p \approx s'\theta}{C'\theta \vee D'\theta \vee [\neg]s\theta \langle t'\theta \rangle|_p \approx s'\theta} \text{ SUP}$$

where $t\theta \approx t'\theta$ is strictly eligible, $[\neg]s\theta \approx s'\theta$ is strictly eligible if positive and eligible if negative, $C\theta \not\leq D\theta$, $t\theta \not\leq t'\theta$ and $s\theta \not\leq s'\theta$. The proof is broken into two cases.

Case 1: $t\theta$ is not at or beneath a variable in C . In this case, p must be a position of s . Let $u = s|_p$. We have that θ is a unifier of u and t and therefore, there must exist an idempotent mgu σ of t and u . The inference conditions can be lifted. $C\theta \not\leq D\theta$ implies $C \not\leq D$, $t\theta \not\leq t'\theta$ implies $t \not\leq t'$ and $s\theta \not\leq s'\theta$ implies $s \not\leq s'$. Moreover $t\theta \approx t'\theta$ being strictly eligible in $D\theta$ implies that $t \approx t'$ is strictly eligible in D and $[\neg]s\theta \approx s'\theta$ being (strictly) eligible in $C\theta$ implies that $[\neg]s \approx s'$ is (strictly) eligible in C . By liftability, we have that the variable condition holds between D and C . Therefore there is the following SUP inference between D and C :

$$\frac{D' \vee t \approx t' \quad C' \vee [\neg]s\theta\langle t \rangle|_p \approx s'}{(C' \vee D' \vee [\neg]s\langle t' \rangle|_p \approx s')\sigma} \text{ SUP}$$

We have $(C' \vee D' \vee [\neg]s\langle t' \rangle|_p \approx s')\sigma\theta = (C' \vee D' \vee [\neg]s\langle t' \rangle|_p \approx s')\theta = C'\theta \vee D'\theta \vee [\neg]s\theta\langle t' \rangle|_p \approx s'\theta$ by the idempotency of σ proving that the conclusion of the ground inference is the θ -ground instance of the conclusion of the non-ground inference and we can take $\theta' = \theta$.

Case 2: $t\theta$ is at or beneath a variable in C which has another instance bad in C . In this case, there must exist positions p' and p'' such that $p = p'.p''$ and $s|_{p'} = y \overline{u_n}$. Let $u = s|_{p'}$. As per case 1, (strict) eligibility of the ground literals implies (strict) eligibility of the non-ground literals. Further $t'\theta$ having a combinator head implies that t' has a variable or combinator head.

If the inference is beneath a variable in C , then by liftability, $y\theta = \tilde{\mathcal{C}} \overline{v_{m>0}}$. If $u = y \overline{u_{n>1}}$, then $u\theta = \tilde{\mathcal{C}} \overline{v_m} (\overline{u_n})\theta$. Thus, $t\theta$ which is a proper subterm of $u\theta$ would be beneath a fully applied combinator which is impossible. Therefore, we have that $n \leq 1$. If the inference is at a variable in C , then we must have $n = 0$ otherwise $y\theta = t\theta$ would not be a first-order subterm.

Thus in both cases $n \leq 1$ and there is a SUBVARSUP inference between C and D :

$$\frac{D' \vee t \approx t' \quad C' \vee [\neg]s\theta\langle y \overline{u_n} \rangle|_{p'} \approx s'}{(C' \vee D' \vee [\neg]s\langle zt' \overline{u_n} \rangle|_{p'} \approx s')\{y \rightarrow zt\}} \text{ SUBVARSUP}$$

where z is a fresh variable. Define the substitution θ' that maps z to $(\lambda x.(y\theta)\langle x \rangle|_{p'})$ and all other variables w to $w\theta$. Since z is fresh, $C\theta = C\theta'$ and $D\theta = D\theta'$. Further, we have that $(zt')\theta' = (\lambda x.(y\theta)\langle x \rangle|_{p'})t'\theta \rightarrow_w^* y\theta\langle t'\theta \rangle|_{p'}$ and $(zt)\theta' \rightarrow_w^* y\theta\langle t\theta \rangle|_{p'} = y\theta$. Therefore:

$$\begin{aligned} & (C' \vee D' \vee [\neg]s\langle zt' \overline{u_n} \rangle|_{p'} \approx s')\{y \rightarrow zt\}\theta' \\ = & (C' \vee D' \vee [\neg]s\langle (\lambda x.(y\theta)\langle x \rangle|_{p'})t' \overline{u_n} \rangle \approx s')\theta[y \rightarrow z\theta't\theta] \\ \rightarrow_w^* & C'\theta \vee D'\theta \vee [\neg]s\theta\langle t'\theta \rangle|_p \approx s'\theta \end{aligned}$$

proving that the conclusion of the ground inference is combinatory congruent to the θ' -instance of the conclusion of the non-ground inference.

Model Construction

In this section the candidate interpretation R_∞ developed in the previous section is shown to be a model of $\lfloor \mathcal{G}_\Sigma(N) \rfloor$. As per the standard proof, this is done by induction on the clause order \succ . For some fixed clause $\lfloor C\theta \rfloor \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$, by Lemma 5, we have that all members of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ are true in $R_{\lfloor C\theta \rfloor}$. By the induction hypothesis, we have that for all clauses $\lfloor D\theta \rfloor \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $C\theta$ are true in $R_{\lfloor C\theta \rfloor}$. In the induction step, it is shown that this implies that $\lfloor C\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$.

Lemma 8. *Let $C\theta, D\theta \in \mathcal{G}_\Sigma(N)$ where the selected literals of $C, D \in N$ correspond with those of $C\theta$ and $D\theta$. Let E be the conclusion of a liftable SUP inference from $C\theta$ and $D\theta$ or an EQRES or EQFACT inference from $C\theta$. Assume that $C\theta$ and $D\theta$ are not redundant with respect to $\mathcal{G}_\Sigma(N)$. Then, $\lfloor E \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta \rfloor$ along with clauses in $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$.*

change the wording, so that only SUP inferences require clauses from $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$.

Proof. By Lemmas 6 and 7, we have that the conclusion of a SUP inference from $C\theta$ and $D\theta$ or an EQRES or EQFACT inference from $C\theta$ is combinatory congruent to the conclusion of the θ' -ground instance of an inference from C and D (or just C as the case may be) where θ' is a ground substitution such that $D\theta' = D\theta$ and $C\theta' = C\theta$. Let E' be the conclusion of the inference between C and D , we have that $E'\theta' \rightarrow_w^* E$. Because N is saturated upto redundancy, the inference is redundant with respect to N . Thus, the θ' ground instance of the inference is redundant with respect to $\mathcal{G}_\Sigma(N)$. By the definition of inference redundancy, this means that $\lfloor E'\theta' \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta' \rfloor = \lfloor C\theta \rfloor$. Using this and the fact that $E'\theta' \rightarrow_w^* E$, we have that $\lfloor E \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta \rfloor$ along with clauses in $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$.

Next, terminology is defined that is used in the following couple of Lemmas and the rest of the paper. Let R be an interpretation and t and t' be ground ceiling terms. Then, $t \sim_R t'$ stands for $R \models \lfloor t \rfloor \approx \lfloor t' \rfloor$. Where the interpretation is obvious, the subscript is omitted.

Lemma 9. *Let R be an interpretation such that every member of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ holds in R . Let u and u' be ground terms such that $u \bar{s} \sim_R u' \bar{s}$ for every type correct tuple of ground terms \bar{s} . Then for ground terms s and s' such that s and s' only differ at positions where s contains u and s' contains u' , we have $s \sim_R s'$.*

Proof. The proof proceeds by showing how equations that are true in R can be used to rewrite s into s' . Since R is a model of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ all ground instances of combinator and extended combinator axioms are true in R .

Let $s_0 = \lfloor s \rfloor$ and $\tilde{s}_0 = \lfloor s' \rfloor$. Terms $s_1, s_2, s_3 \dots$ are defined inductively as follows: s_{i+1} is formed from s_i by rewriting all subterms of the form $\lfloor u \bar{v} \rfloor$ in s_i to $\lfloor u' \bar{v} \rfloor$ and then reducing the outermost leftmost weak-redex in the resulting term. Terms $\tilde{s}_1, \tilde{s}_2, \tilde{s}_3 \dots$ are also defined inductively: \tilde{s}_{i+1} is formed from \tilde{s}_i by reducing the left-most outermost weak-redex in \tilde{s}_i .

The algorithm maintains the invariant that for all i , $\lceil s_i \rceil$ and $\lceil \tilde{s}_i \rceil$ are identical other than at positions where $\lceil s_i \rceil$ contains a u and $\lceil \tilde{s}_i \rceil$ contains a u' . Let \tilde{s}_* be the final

term in the \tilde{s}_n series. Such a term must exist as weak reduction is terminating. Let s_* be the equivalent term in s_n series. We show that $s_* = \tilde{s}_*$.

Assume that $s_* \neq \tilde{s}_*$. Consider the outermost position at which $\lceil s_* \rceil$ and $\lceil \tilde{s}_* \rceil$ differ. If this position is not beneath a fully applied combinator in $\lceil s_* \rceil$, then s_* and \tilde{s}_* must contain $\lfloor u \bar{v} \rfloor$ and $\lfloor u' \bar{v}' \rfloor$ at the corresponding positions. But this is impossible since $\lfloor u \bar{v} \rfloor$ would have been rewritten to $\lfloor u' \bar{v}' \rfloor$ at the previous step of the algorithm. Therefore, assume that the outermost position at which $\lceil s_* \rceil$ and $\lceil \tilde{s}_* \rceil$ differ is beneath a fully applied combinator in $\lceil s_* \rceil$. Since we are considering the outermost position, this combinator cannot be a part of u . Thus, we must have that the same combinator occurs fully applied in $\lceil \tilde{s}_* \rceil$. This contradicts the fact that \tilde{s}_* is in weak normal form. Therefore the original assumption is false and $s_* = \tilde{s}_*$.

Lemma 10. *Let R be an interpretation such that every member of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ holds in R . Let u and u' be ground terms such that $u \sim_R u'$. Let $s[\]_n$ be a ground context with n holes at stable positions. Then $s[\bar{u}]_n \sim_R s[\bar{u}']_n$.*

Proof. The proof proceeds by induction on $\|s[\]_n\|$. In the base case $\|s[\]_n\| = 0$ and no instance of u occurs beneath a fully applied combinator. By the definition of stable positions, no instance of u occurs with arguments. Thus, $\lfloor s[\bar{u}]_n \rfloor$ can be rewritten to $\lfloor s[\bar{u}']_n \rfloor$ directly using the equation $\lfloor u \approx u' \rfloor$.

For the inductive case, we have that $\|s[\]_n\| > 0$. This splits into two cases depending on whether any of the holes are underneath the leftmost fully applied combinator or not.

Case 1: There are no holes underneath the leftmost fully applied combinator. Consider the context $s'[\]_n$ formed by reducing the leftmost fully applied combinator in s . Because $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ holds in R , we have $s[\bar{u}]_n \sim_R s'[\bar{u}]_n$. By the induction hypothesis we have $s'[\bar{u}]_n \sim_R s'[\bar{u}']_n$ and then using the relevant member of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$, we have $s'[\bar{u}']_n \sim_R s[\bar{u}']_n$. By the transitivity of \sim_R , we have $s[\bar{u}]_n \sim_R s[\bar{u}']_n$.

Case 2: The leftmost fully applied combinator is of the form $\tilde{\mathcal{C}} \bar{t}_k$ where at least one t_i has a hole as a proper subterm. Note that by the definition of stable subterms, none of the t_i s can be a hole. Consider the context $s'[\]_m$ formed by replacing $\tilde{\mathcal{C}} \bar{t}_k$ with $(\tilde{\mathcal{C}} \bar{t}_k) \downarrow^w$ in s . Because any holes occurring as subterms of $\tilde{\mathcal{C}} \bar{t}_k$ occur at stable positions, any holes occurring in $(\tilde{\mathcal{C}} \bar{t}_k) \downarrow^w$ also occur at stable positions. Thus all holes in $s'[\]_m$ occur at stable positions. Because $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ holds in R , we have $s[\bar{u}]_n \sim_R s'[\bar{u}]_m$. By the induction hypothesis we have $s'[\bar{u}]_m \sim_R s'[\bar{u}']_m$ and then using the relevant member of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$, we have $s'[\bar{u}']_m \sim_R s[\bar{u}']_n$. By the transitivity of \sim_R , we have $s[\bar{u}]_n \sim_R s[\bar{u}']_n$.

Lemma 11. *Let $C\theta, D\theta \in \mathcal{G}_\Sigma(N)$, where the selected literals of $C\theta = C'\theta \vee [\neg]s\theta \approx s'\theta$ and $D\theta = D'\theta \vee t\theta \approx t'\theta$ match those of $C, D \in N$. Consider a non-liftable SUP inference between $C\theta$ and $D\theta$. Assume that $C\theta$ and $D\theta$ are not redundant with respect to $\mathcal{G}_\Sigma(N)$. Then $\lfloor C\theta \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than it, clauses in $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ and $\neg \lfloor D'\theta \rfloor$.*

Proof. Assume that the $s\theta \succ s'\theta$ and $t\theta \succ t'\theta$. Let R be an interpretation such that clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta \rfloor$ and all members of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ are true in R whilst $\lfloor D'\theta \rfloor$ is false. By the SUP conditions, we have that either $C\theta \succ D\theta$ or

define multi-hole contexts

$D\theta \in \mathcal{G}_\Sigma(ECA)$ and in both cases $R \models [t\theta \approx t'\theta]$. We need to show that this implies $R \models [C\theta]$.

The inference can be non-liftable for one of the following reasons:

1. The variable condition not holding between C and D ;
2. It is at or below a variable in C all of whose other occurrences are good in C ;
3. It is below a variable x in C and $x\theta$ has a first-order head;
4. It is at or below a variable in C and $t'\theta$ has a first-order head;
5. It is at or below a variable in C and D is an extended combinator axiom;

We fix some terminology common to cases 2-5. Let x be the variable at or beneath which the inference takes place. Then $t\theta$ is a first-order subterm of $x\theta$. Let v be the result of replacing $t\theta$ by $t'\theta$ in $x\theta$ at the relevant position. Let $C' = C\theta[x \rightarrow v]$.

Case 1: The variable condition fails to hold because D is an extended combinator axiom not compatible with u . By the definition of compatibility, $u = x \overline{s_m}$ and $t = \tilde{C} \overline{x_{n-m}} \approx t'$ and $\tilde{C} \overline{x_{n-m}}$ is a fully applied combinator. Thus, $x\theta = (\tilde{C} \overline{x_{n-m}})\theta$ is also a fully applied combinator. Let $C'' = C\{x \mapsto ((\tilde{C} \overline{x_{n-m}})\theta) \downarrow^w\}$. Because the maximal weak reduction from the largest term in $C\theta$ is greater than the maximal weak reduction from the largest term in $C''\theta$, we have $C\theta \succ C''\theta$ and thus $R \models [C''\theta]$. But then, as $C\theta \rightarrow_w C''\theta$ and all members of $[\mathcal{G}_\Sigma(ECA)]$ are true in R , we have $R \models [C\theta]$ by congruence.

Case 2: The superposed subterm $t\theta$ is at or beneath a variable x in C all of whose other occurrences are good in C . From the existence of the inference between $C\theta$ and $D\theta$, it follows that $t\theta$ must occur at a first-order and thus stable position in $x\theta$. Likewise, the existence of the inference implies that the $x\theta$ involved in the inference occurs at a stable position within $s\theta$. For all other instances of $x\theta$, the fact that x is good in C implies that $x\theta$ is stable in $C\theta$. Thus, by compatibility with stable contexts and the fact that $t\theta \succ t'\theta$, we have that $C\theta \succ C'$ and thus by assumption on R , $R \models [C']$. Since $C\theta$ and C' only differ at stable positions where one contains $t\theta$ and the other $t'\theta$, Lemma 10 can be used to show that every literal of $C\theta$ is equivalent in R to the corresponding literal of C' . This implies $R \models [C\theta]$ by congruence.

Case 3: The superposed subterm $t\theta$ is beneath a variable x in C and $x\theta$ has a first order head. We have that $x\theta = (f \overline{s_n})\langle t\theta \rangle$ and $v = (f \overline{s_n})\langle t'\theta \rangle$. Thus, $x\theta \succ v$ follows from the ordering's compatibility with stable context. As v has a first order head and C' is formed from $C\theta$ by replacing $x\theta$ with v , $C\theta \succ C'$ follows from property P2 of the ordering. Hence, $R \models [C']$ and $R \models [C\theta]$ via Lemma 9 and congruence.

Case 4: The superposed subterm $t\theta$ is at or beneath a variable x in C and $t'\theta$ has a first order head. We have $t\theta \succ t'\theta$ and C' can be formed from $C\theta$ by replacing all occurrences of t with t' . Thus, $C\theta \succ C'$ follows from property P2 of the ordering. Hence, $R \models [C']$.

We perform a case analysis depending on whether $[\neg]s\theta \approx s'\theta$ is a maximal or selected literal in $C\theta$. First though, note that for all type correct tuple of ground terms \overline{u} , if $t\theta \overline{u}$ and $t'\theta \overline{u}$ are smaller than the maximal term of $C\theta$ then:

$$R \models [t\theta \overline{u} \approx t'\theta \overline{u}] \quad (1)$$

This can be shown exactly as Bentkamp et al. do so [1]. If $[\neg]s\theta \approx s'\theta$ is the maximal literal in $C\theta$, consider the clause C'' formed by rewriting $[t\theta]$ to $[t'\theta]$ wherever

possible in $\lfloor C\theta \rfloor$. As $t\theta$ is a first-order subterm of the maximal term of $C\theta$, $s\theta$, we have that every term of C'' is smaller than the maximal term of $\lfloor C\theta \rfloor$. Further, we have that $\lceil C'' \rceil$ and C' differ only at positions where $\lceil C'' \rceil$ contains a $t\theta$ and C' contains a $t'\theta$. We prove that C'' can be rewritten to $\lfloor C' \rfloor$ using equalities true in R . Hence $R \models \lfloor C\theta \rfloor$ via congruence.

Let $l \approx r$ be an arbitrary literal of C'' and $l' \approx r'$ be the corresponding literal in $\lfloor C' \rfloor$. Then, since $\lceil C'' \rceil$ and C' only differ where $\lceil C'' \rceil$ contains a $t\theta$ and C' contains a $t'\theta$, we have that $l = \lfloor k[\overline{t\theta}_n] \rfloor$ and $l' = \lfloor k[\overline{t'\theta}_n] \rfloor$ for some context k . We provide an algorithm for rewriting l into l' using equalities true in R . The same can be done for r and r' proving that the literal $l \approx r$ can be rewritten to the literal $l' \approx r'$. Since the literal was chosen arbitrarily, this proves that every literal of C'' can be rewritten to the corresponding literal of $\lfloor C' \rfloor$ in R .

The algorithm is the same as that in Lemma 9. It is repeated here for clarity. Let $l_0 = \lfloor l \rfloor$ and $\tilde{l}_0 = \lfloor l' \rfloor$. Terms $l_1, l_2, l_3 \dots$ are defined inductively as follows: l_{i+1} is formed from l_i by rewriting all subterms of the form $\lfloor t\theta \overline{u} \rfloor$ in l_i to $\lfloor t'\theta \overline{u} \rfloor$ and then reducing the outermost leftmost weak-redex in the resulting term. Terms $\tilde{l}_1, \tilde{l}_2, \tilde{l}_3 \dots$ are also defined inductively: \tilde{l}_{i+1} is formed from \tilde{l}_i by reducing the left-most outermost weak-redex in \tilde{l}_i .

The argument that the algorithm terminates and that l_* and \tilde{l}_* are syntactically identical is the same as for Lemma 9. The reason Lemma 9 cannot be invoked here is that using Lemma 9 would require that $R \models \lfloor t\theta \overline{w} \approx t'\theta \overline{w} \rfloor$ for every type correct tuple of terms \overline{w} . In the current context, this does not hold. It therefore remains to justify the rewrites from each l_i to l_{i+1} in the above algorithm. Both l_0 and \tilde{l}_0 are smaller than $\lfloor s\theta \rfloor$. Reducing a leftmost redex of a term results in a smaller term. Likewise, by property P2 of the ordering, rewriting a term of the form $\lfloor t\theta \overline{u} \rfloor$ to one of the form $\lfloor t'\theta \overline{u} \rfloor$ results in a smaller term. Therefore, for all $i > 0$, $l_i \prec l_{i-1} \prec \dots \prec l_0 \prec \lfloor s\theta \rfloor$ justifying the use of Equation 1.

If $\lceil \neg \rceil s\theta \approx s'\theta$ is a selected literal, then by the selection criteria x cannot be the head of the maximal term of C . Therefore, the algorithm provided above can be used to rewrite C'' into $\lfloor C' \rfloor$ proving that $R \models C\theta$ by congruence.

Case 5: The superposed subterm $t\theta$ is at or beneath a variable x in C and $t\theta \approx t'\theta$ is a member of $\mathcal{G}_\Sigma(ECA)$. In this case $C\theta \rightarrow_w C'$ and thus $C\theta \succ C'$ by property P1 of the ordering. Hence, $R \models \lfloor C' \rfloor$. Since all members of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ are true in R , every side of a literal in $C\theta$ is equal in R to the equivalent term in C' and $R \models C\theta$ follows by congruence.

Lemma 12 (R_∞ is a model). Let $\lfloor C\theta \rfloor \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$, then

- (i) $E_{\lfloor C\theta \rfloor} = \emptyset$ if and only if $R_{\lfloor C\theta \rfloor} \models C\theta$;
- (ii) if $C\theta$ is redundant with respect to $\mathcal{G}_\Sigma(N)$ then $\lfloor C\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$;
- (iii) $\lfloor C\theta \rfloor$ holds in R_∞ and R_D for all $D \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$, $D \succ C\theta$;
- (iv) if $C\theta$ has selected literals, then $R_{\lfloor C\theta \rfloor} \models \lfloor C\theta \rfloor$;
- (v) if $C\theta$ is a member of $\mathcal{G}_\Sigma(ECA)$, then $R_{\lfloor C\theta \rfloor} \models \lfloor C\theta \rfloor$.

Proof. The proof proceeds by induction on ground clauses of the floor logic. We assume that (i) to (v) are satisfied by all clauses $D \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$ such that $\lfloor C\theta \rfloor \succ D$. We prove that (i) to (v) hold for $C\theta$. The \Rightarrow direction of (i) follows from the construction. Part

(iii) follows from (i) by Lemmas 3 and 4. Part (v) is a straightforward extension of 5. Therefore it remains to prove the \Leftarrow direction of (i) and (ii) and (iv) for the case where $C\theta \notin \mathcal{G}_\Sigma(ECA)$. We assume that the selected literals of $C\theta$ match those of $C \in N$.

Case 1: $C\theta$ is redundant with respect to $\lfloor \mathcal{G}_\Sigma(N) \rfloor$. Then $\lfloor C\theta \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ that are smaller than it. By part (iii) of the induction hypothesis, these clauses are true in $R_{\lfloor C\theta \rfloor}$ and therefore $\lfloor C\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$.

Case 2: $C\theta = C'\theta \vee s\theta \not\approx s'\theta$ and $C\theta$ is not redundant with respect to $\mathcal{G}_\Sigma(N)$.

Case 2.1: $s\theta = s'\theta$. In this case, there is an EQRES inference from $C\theta$:

$$\frac{C'\theta \vee s\theta \not\approx s'\theta}{C'\theta} \text{EQRES}$$

by Lemma 8, we have that $\lfloor C'\theta \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta \rfloor$. By part (iii) of the induction hypothesis, these clauses are true in $R_{\lfloor C\theta \rfloor}$. Therefore, $\lfloor C'\theta \rfloor$ and thus $\lfloor C\theta \rfloor$ are true in $R_{\lfloor C\theta \rfloor}$.

Case 2.2: $s\theta \succ s'\theta$. If $R_{\lfloor C\theta \rfloor} \models s\theta \not\approx s'\theta$ then the lemma follows. Therefore, assume that it doesn't hold and $s\theta \downarrow_{R_{\lfloor C\theta \rfloor}} s'\theta$. There must exist some rule in $R_{\lfloor C\theta \rfloor}$ which reduces $s\theta$. Such a rule must have been produced by some clause $\lfloor D\theta \rfloor = \lfloor D'\theta \vee t\theta \approx t'\theta \rfloor$. Without loss of generality, it is assumed that the C and D are variable disjoint, so that the same substitution θ can be used. It is also assumed that the selected literals of $D\theta$ match those of D . Then there exists the following SUP inference:

$$\frac{D'\theta \vee t\theta \approx t'\theta \quad C'\theta \vee s\theta \langle t\theta \rangle \not\approx s'\theta}{C'\theta \vee D'\theta \vee s\theta \langle t'\theta \rangle \not\approx s'\theta} \text{SUP}$$

by Lemma 11, if the inference is non-liftable then $\lfloor C\theta \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ that are smaller than it, by $\neg \lfloor D'\theta \rfloor$ and by clauses in $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$. By part (iii) of the induction hypothesis, clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $C\theta$ are true in $R_{\lfloor C\theta \rfloor}$. By Lemma 5 all members of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ are true in $R_{\lfloor C\theta \rfloor}$. By Lemma 4 $\neg \lfloor D'\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$. Therefore $C\theta$ is true in $R_{\lfloor C\theta \rfloor}$.

If the inference is liftable then by Lemma 8, we have that $\lfloor C'\theta \vee D'\theta \vee s\theta \langle t'\theta \rangle \not\approx s'\theta \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta \rfloor$ along with clauses in $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$. By part (iii) of the induction hypothesis and Lemma 5, all these clauses are true in $R_{\lfloor C\theta \rfloor}$. Therefore, $\lfloor C'\theta \vee D'\theta \vee s\theta \langle t'\theta \rangle \not\approx s'\theta \rfloor$ holds in $R_{\lfloor C\theta \rfloor}$. Since $\lfloor D'\theta \rfloor$ is false in $R_{\lfloor C\theta \rfloor}$, either $R_{\lfloor C\theta \rfloor} \models \lfloor C'\theta \rfloor$ or $R_{\lfloor C\theta \rfloor} \models \lfloor s\theta \langle t'\theta \rangle \not\approx s'\theta \rfloor$. In the latter case, $R_{\lfloor C\theta \rfloor} \models \lfloor s\theta \langle t\theta \rangle \not\approx s'\theta \rfloor$ because $\lfloor t\theta \rfloor \rightarrow \lfloor t'\theta \rfloor \in R_{\lfloor C\theta \rfloor}$. Thus, in both cases $R_{\lfloor C\theta \rfloor} \models \lfloor C\theta \rfloor$.

Case 3: $C\theta$ is not redundant and contains no negative literals. In this case $C\theta = C'\theta \vee s\theta \approx s'\theta$. If $E_{\lfloor C\theta \rfloor} = \{\lfloor s\theta \rfloor \rightarrow \lfloor s'\theta \rfloor\}$ or $R_{\lfloor C\theta \rfloor} \models C'\theta$ or $s\theta = s'\theta$ then $R_{\lfloor C\theta \rfloor} \models C\theta$. Therefore, assume $E_{\lfloor C\theta \rfloor} = \emptyset$, $s\theta \neq s'\theta$ and $C'\theta$ is false in $R_{\lfloor C\theta \rfloor}$.

Case 3.1: $\lfloor s\theta \approx s'\theta \rfloor$ is not strictly maximal in $C\theta$. In this case, $C\theta$ can be written as $C''\theta \vee t\theta \approx t'\theta \vee s\theta \approx s'\theta$ where $t\theta = s\theta$ and $t'\theta = s'\theta$. Then there is an EQFACT inference from $C\theta$:

$$\frac{C''\theta \vee t\theta \approx t'\theta \vee s\theta \approx s'\theta}{C''\theta \vee t'\theta \not\approx s'\theta \vee t\theta \approx t'\theta} \text{EQFACT}$$

by Lemma 8 the conclusion of the inference is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta \rfloor$ and therefore by part (iii) of the induction hypothesis true in $R_{\lfloor C\theta \rfloor}$. Since $t'\theta = s'\theta$, $\lfloor t'\theta \not\approx s'\theta \rfloor$ is false in $R_{\lfloor C\theta \rfloor}$. Therefore $\lfloor t\theta \approx t'\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$ and hence $\lfloor C\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$.

Case 3.2: The literal $s\theta \approx s'\theta$ is strictly maximal and $\lfloor s\theta \rfloor$ is reducible by some rule in $R_{\lfloor C\theta \rfloor}$. This rule must be produced by a clause $\lfloor D\theta \rfloor = \lfloor D'\theta \vee t\theta \approx t'\theta \rfloor$. Without loss of generality, we assume that D and C are variable disjoint so that the same substitution θ can be used. We also assume that the selected literals of D match those of $D\theta$. then there is an SUP inference:

$$\frac{D'\theta \vee t\theta \approx t'\theta \quad C'\theta \vee s\theta \langle t\theta \rangle \approx s'\theta}{C'\theta \vee D'\theta \vee s\theta \langle t'\theta \rangle \approx s'\theta} \text{ SUP}$$

by Lemma 11, if the inference is non-liftable then $\lfloor C\theta \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ that are smaller than it, by $\neg \lfloor D'\theta \rfloor$ and by clauses in $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$. By part (iii) of the induction hypothesis, clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $C\theta$ are true in $R_{\lfloor C\theta \rfloor}$. By Lemma 5 all members of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ are true in $R_{\lfloor C\theta \rfloor}$. By Lemma 4 $\neg \lfloor D'\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$. Therefore $C\theta$ is true in $R_{\lfloor C\theta \rfloor}$.

If the inference is liftable then by Lemma 8, we have that $\lfloor C'\theta \vee D'\theta \vee s\theta \langle t'\theta \rangle \approx s'\theta \rfloor$ is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta \rfloor$ along with clauses in $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$. By part (iii) of the induction hypothesis and Lemma 5, all these clauses are true in $R_{\lfloor C\theta \rfloor}$. Therefore, $\lfloor C'\theta \vee D'\theta \vee s\theta \langle t'\theta \rangle \approx s'\theta \rfloor$ holds in $R_{\lfloor C\theta \rfloor}$. Since $\lfloor D'\theta \rfloor$ is false in $R_{\lfloor C\theta \rfloor}$, either $R_{\lfloor C\theta \rfloor} \models \lfloor C'\theta \rfloor$ or $R_{\lfloor C\theta \rfloor} \models \lfloor s\theta \langle t'\theta \rangle \approx s'\theta \rfloor$. In the latter case, $R_{\lfloor C\theta \rfloor} \models \lfloor s\theta \langle t\theta \rangle \approx s'\theta \rfloor$ because $\lfloor t\theta \rangle \rightarrow \lfloor t'\theta \rangle \in R_{\lfloor C\theta \rfloor}$. Thus, in both cases $R_{\lfloor C\theta \rfloor} \models \lfloor C\theta \rfloor$.

Case 3.3: $s\theta \approx s'\theta$ is strictly maximal and $\lfloor s\theta \rfloor$ is not reducible by any rule in $R_{\lfloor C\theta \rfloor}$. By assumption we have that $E_{\lfloor C\theta \rfloor} = \emptyset$. Assume that $\lfloor C\theta \rfloor$ is false in $R_{\lfloor C\theta \rfloor}$. By the construction of $E_{\lfloor C\theta \rfloor}$, it must be the case that $C'\theta$ is true in $R_{\lfloor C\theta \rfloor} \cup \{\lfloor s\theta \rangle \rightarrow \lfloor s'\theta \rangle\}$. Thus $C'\theta$ must have the form $C''\theta \vee t\theta \approx t'\theta$ where the literal $\lfloor t\theta \approx t'\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor} \cup \{\lfloor s\theta \rangle \rightarrow \lfloor s'\theta \rangle\}$, but not in $R_{\lfloor C\theta \rfloor}$. By the confluence of $R_{\lfloor C\theta \rfloor}$, this is equivalent to saying $t\theta \downarrow_{R_{\lfloor C\theta \rfloor} \cup \{\lfloor s\theta \rangle \rightarrow \lfloor s'\theta \rangle\}} t'\theta$, but not $t\theta \downarrow_{R_{\lfloor C\theta \rfloor}} t'\theta$. Therefore, the rule $\lfloor s\theta \rangle \rightarrow \lfloor s'\theta \rangle$ must be used at least once in rewriting either $t\theta$ or $t'\theta$ to a normal form. As $s\theta \succ s'\theta$, $t\theta \succ t'\theta$ and $s\theta \approx s'\theta \succ t\theta \approx t'\theta$, we have that $s\theta \succ t'\theta$ and $s\theta \succeq t\theta$. But the fact that $\lfloor s\theta \rangle \rightarrow \lfloor s'\theta \rangle$ is used in the rewrite proof implies that $s\theta = t\theta$ and that the rewrite proof looks like this: $\lfloor t\theta \rangle \rightarrow \lfloor s'\theta \rangle \rightarrow^* u \leftarrow^* \lfloor t'\theta \rangle$. Hence, we have that $R_{\lfloor C\theta \rfloor} \models \lfloor s'\theta \approx t'\theta \rfloor$. Now consider the following EQFACT inference from $C\theta$:

$$\frac{C''\theta \vee t\theta \approx t'\theta \vee s\theta \approx s'\theta}{C''\theta \vee t'\theta \not\approx s'\theta \vee t\theta \approx t'\theta} \text{ EQFACT}$$

by Lemma 8 the conclusion of the inference is entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ smaller than $\lfloor C\theta \rfloor$ and therefore by part (iii) of the induction hypothesis true in $R_{\lfloor C\theta \rfloor}$. Since $\lfloor t'\theta \not\approx s'\theta \rfloor$ is false in $R_{\lfloor C\theta \rfloor}$, $\lfloor t\theta \approx t'\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$ and hence $\lfloor C\theta \rfloor$ is true in $R_{\lfloor C\theta \rfloor}$ contradicting our assumption.

is this assumption really required?

7 Construction of Higher-order Model

Lemma 13. *For terms, t, t' and u , if $t \sim t'$ then $tu \sim t'u$*

Proof. The proof proceeds by induction on the number of rewrite steps between $\lfloor t \rfloor$ and $\lfloor t' \rfloor$. If $t = t'$ then the Lemma follows trivially. Let the number of rewrite steps between $\lfloor t \rfloor$ and $\lfloor t' \rfloor$ be n . Let t'' be the term such that $\lfloor t' \rfloor$ rewrites to $\lfloor t'' \rfloor$ in $n - 1$ steps and $\lfloor t'' \rfloor$ rewrites to $\lfloor t \rfloor$ in a single step. By the induction hypothesis we have that $t'u \sim t''u$. Thus, if it can be shown that $t''u \sim tu$ the Lemma follows by the transitivity of \sim .

Since $\lfloor t'' \rfloor$ rewrites to $\lfloor t \rfloor$ in a single step, it must be the case that either $\lfloor t'' \rfloor \rightarrow \lfloor t \rfloor$ or $\lfloor t \rfloor \rightarrow \lfloor t'' \rfloor$ is a rule in R_∞ . Without loss of generality, assume that it is the first. This rule must stem from a productive clause of the form $\lfloor C\theta \rfloor \approx \lfloor C'\theta \vee t_1\theta \approx t_2\theta \rfloor$ where $t_1\theta \approx t''$ and $t_2\theta \approx t$. By the definition of a productive clause $t'' \approx t$ is strictly eligible in $C\theta$ and likewise $t_1 \approx t_2$ is strictly eligible in C . Further, t'' and t are of functional type, so t_1 and t_2 must be of functional or polymorphic type. Thus, there is an ARGCONG inference from C with conclusion $(C' \vee t_1x \approx t_2x)\sigma$. Call this conclusion E .

By part (ii) of Lemma 12, $C\theta$ is not redundant and therefore C is not redundant. Thus E is either in N or $Red(N)$. The ground instance of $\lfloor E \rfloor$, $\lfloor C'\theta \vee t''u \approx tu \rfloor$ is thus either in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$ or entailed by clauses in $\lfloor \mathcal{G}_\Sigma(N) \rfloor$. Therefore it is true in R_∞ . By Lemma 4, we have that $\lfloor C'\theta \rfloor$ is false in R_∞ which implies that $\lfloor t''u \approx tu \rfloor$ must be true.

Lemma 14. *For all ground terms t, t', u, u' , if $t \sim t'$ and $u \sim u'$, then $tu \sim t'u'$ if neither t nor t' is of the form $\mathcal{C} \overline{s_{n>1}}, \mathbf{K} \overline{s_{n>0}}$ or $\mathbf{I} \overline{s_n}$.*

Proof. By Lemma 13, we have that $t'u' \sim tu'$, so if it can be shown that $tu' \sim tu$ the Lemma follows immediately by the transitivity of \sim . We have that $\lfloor t \rfloor = \zeta_n(\overline{s_n})$. By the condition on the form of t and t' , tu' cannot have a fully applied combinator as its head symbol. Therefore, $\lfloor tu' \rfloor = \zeta_{n+1}(\overline{s_n}, u')$. It is obvious that any rewrite steps from $\lfloor u' \rfloor$ can be carried out from $\lfloor tu' \rfloor$ and therefore $tu' \sim tu$.

Lemma 15. *For all ground terms u and u' such that $u \sim u'$, $\mathcal{C} t_1 t_2 u \sim \mathcal{C} t_1 t_2 u'$ and $\mathbf{K} t u \sim \mathbf{K} t u'$ and $\mathbf{I} u \sim \mathbf{I} u'$.*

Proof. By multiple applications of Lemma 13, we have that $\lfloor u \overline{t} \rfloor \approx \lfloor u' \overline{t} \rfloor$ holds in R_∞ for all type correct tuple of terms \overline{t} . We also have that every member of $\lfloor \mathcal{G}_\Sigma(ECA) \rfloor$ holds in R_∞ . Thus, the lemma follows by an appeal to Lemma 9 with $R = R_\infty$.

Theorem 1. *For all ground terms t, t', u and u' , if $t \sim t'$ and $u \sim u'$, then $tu \sim t'u'$.*

Proof. Proof is by induction on $\|t\| + \|u\| + \|t'\| + \|u'\|$. the base case splits into two cases.

Case 1: Neither t nor t' is of the form $\mathcal{C} t_1 t_2$, $\mathbf{K} t$ or \mathbf{I} . Then the proof follows by an application of Lemma 13.

Case 2: One or both of t and t' are of the form $\mathcal{C} t_1 t_2$, $\mathbf{K} t$ or \mathbf{I} . Without loss of generality, assume that t is of the form $\mathcal{C} t_1 t_2$. By Lemma 14, $t'u' \sim tu'$. Thus, if it can be proven that $tu' = \mathcal{C} t_1 t_2 u' \sim \mathcal{C} t_1 t_2 u = tu$, the theorem follows by the transitivity of \sim . By Lemma 15, $\mathcal{C} t_1 t_2 u' \sim \mathcal{C} t_1 t_2 u$ completing the base case.

and parts (i) and (iv)?

For the inductive case, one or more of $\|t\|$, $\|u\|$, $\|t'\|$ or $\|u'\|$ is greater than 0. We show that the theorem holds for the first two cases. The latter two can be proved in a like manner.

Case 1: $\|t\| > 0$. Let $t'' = (t) \downarrow^w$. Since R_∞ is a model of $\lfloor \mathcal{G}_\Sigma(N) \rfloor$, the floors of the ground instances of all combinator and extended combinator axioms must be true in R_∞ . Thus, $t \sim t''$ and by Lemma 13, $tu \sim t''u$. Since $\|t''\| < \|t\|$, the induction hypothesis can be used to conclude that $t''u \sim t'u'$. By the transitivity of \sim , $tu \sim t'u'$ follows.

Case 2: $\|u\| > 0$. Since R_∞ models all combinator and extended combinator axioms, we have $u \sim u_2$ where $u_2 = (u) \downarrow^w$. By the induction hypothesis, we have $tu_2 \sim t'u'$. Either Lemma 14 or Lemma 15 is applicable to prove $tu \sim tu_2$. The theorem follows by the transitivity of \sim .

Definition 3. Define an interpretation $R_\infty^\uparrow = (\mathcal{U}^\uparrow, \mathcal{E}^\uparrow, \mathcal{J}^\uparrow)$ in the ceiling logic as follows. Let $(\mathcal{U}, \mathcal{E}, \mathcal{J}) = R_\infty$. Let $\mathcal{U}_\tau^\uparrow = \mathcal{U}_{\lfloor \tau \rfloor}$ and $\mathcal{J}^\uparrow(f, \bar{\tau}) = \mathcal{J}(f_0^\tau, \lfloor \bar{\tau} \rfloor)$. Since R_∞ is term-generated, for every $a \in \mathcal{U}_{\lfloor \tau \rightarrow v \rfloor}$ and $b \in \mathcal{U}_{\lfloor \tau \rfloor}$, there exists ground ceiling terms $s : \tau \rightarrow v$ and $t : \tau$ such that $\llbracket s \rrbracket_{R_\infty}^\xi = a$ and $\llbracket t \rrbracket_{R_\infty}^\xi = b$.

Define \mathcal{E}^\uparrow by

$$\mathcal{E}_{\tau, v}^\uparrow(a)(b) = \llbracket st \rrbracket_{R_\infty}^\xi$$

This interpretation is well defined if the definition of \mathcal{E}^\uparrow does not depend on the choice of the ground terms s and t . To show this, we assume that there exists other ground terms s' and t' such that $\llbracket s' \rrbracket_{R_\infty}^\xi = a$ and $\llbracket t' \rrbracket_{R_\infty}^\xi = b$. By Theorem 1, it follows from $\llbracket s \rrbracket_{R_\infty}^\xi = \llbracket s' \rrbracket_{R_\infty}^\xi$ and $\llbracket t \rrbracket_{R_\infty}^\xi = \llbracket t' \rrbracket_{R_\infty}^\xi$ that

$$\llbracket st \rrbracket_{R_\infty}^\xi = \llbracket s't' \rrbracket_{R_\infty}^\xi$$

indicating that the definition of \mathcal{E}^\uparrow is independent of the choice of s and t .

Since R_∞ is a term-generated model of $\lfloor \mathcal{G}_\Sigma(N) \rfloor$, we can show that R_∞^\uparrow is also term-generated.

Lemma 16 (Substitution lemma). For all ceiling logic terms t and grounding substitutions ρ , $\llbracket t\rho \rrbracket_{R_\infty^\uparrow}^\xi = \llbracket t \rrbracket_{R_\infty^\uparrow}^\xi$ if $\alpha\xi = \alpha\rho$ for all α and $\xi(x) = \llbracket x\rho \rrbracket_{R_\infty^\uparrow}^\xi$ for all x .

Proof. By induction on the structure of t . If t is a variable x , then $\llbracket x \rrbracket_{R_\infty^\uparrow}^\xi = \xi(x) = \llbracket x\rho \rrbracket_{R_\infty^\uparrow}^\xi$.

If t is of the form $f(\bar{\tau})$, then $\llbracket t \rrbracket_{R_\infty^\uparrow}^\xi = \mathcal{J}^\uparrow(f, \llbracket \bar{\tau} \rrbracket_{R_\infty^\uparrow}^\xi) = \mathcal{J}^\uparrow(f, \bar{\tau}\rho) = \llbracket f(\bar{\tau})\rho \rrbracket_{R_\infty^\uparrow}^\xi$.

Finally if t is an application of the form $t_1 t_2$, then:

$$\begin{aligned} \llbracket t_1 t_2 \rrbracket_{R_\infty^\uparrow}^\xi &= \mathcal{E}^\uparrow(\llbracket t_1 \rrbracket_{R_\infty^\uparrow}^\xi)(\llbracket t_2 \rrbracket_{R_\infty^\uparrow}^\xi) \\ &\stackrel{\text{IH}}{=} \mathcal{E}^\uparrow(\llbracket t_1\rho \rrbracket_{R_\infty^\uparrow}^\xi)(\llbracket t_2\rho \rrbracket_{R_\infty^\uparrow}^\xi) \\ &= \llbracket t_1\rho t_2\rho \rrbracket_{R_\infty^\uparrow}^\xi \\ &= \llbracket (t_1 t_2)\rho \rrbracket_{R_\infty^\uparrow}^\xi \end{aligned}$$

Lemma 17 (Model transfer to ceiling logic). R_∞^\uparrow is a term-generated model of $\mathcal{G}_\Sigma(N)$.

Proof. By induction on ground terms t of the ceiling logic it is shown that $\llbracket t \rrbracket_{R_\infty^\uparrow}^\xi = \llbracket \lfloor t \rfloor \rrbracket_{R_\infty}^\xi$. Let t be a ground ceiling term. If t is of the form $f\langle \bar{\tau} \rangle$, then $\llbracket t \rrbracket_{R_\infty^\uparrow}^\xi = \mathcal{J}^\uparrow(f, \llbracket \bar{\tau} \rrbracket^\xi) = \mathcal{J}^\uparrow(f, \bar{\tau}) = \mathcal{J}(f_0, \lfloor \bar{\tau} \rfloor) = \mathcal{J}(f_0, \llbracket \lfloor \bar{\tau} \rfloor \rrbracket^\xi) = \llbracket \lfloor t \rfloor \rrbracket_{R_\infty}^\xi$.

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

$$\begin{aligned} \llbracket t_1 t_2 \rrbracket_{R_\infty^\uparrow}^\xi &= \mathcal{E}_{\tau, v}^\uparrow(\llbracket t_1 \rrbracket_{R_\infty^\uparrow}^\xi)(\llbracket t_2 \rrbracket_{R_\infty^\uparrow}^\xi) \\ &\stackrel{\text{IH}}{=} \mathcal{E}_{\tau, v}^\uparrow(\llbracket \lfloor t_1 \rfloor \rrbracket_{R_\infty}^\xi)(\llbracket \lfloor t_2 \rfloor \rrbracket_{R_\infty}^\xi) \\ &\stackrel{\text{Def } \mathcal{E}^\uparrow}{=} \llbracket \lfloor t_1 t_2 \rfloor \rrbracket_{R_\infty}^\xi \end{aligned}$$

We have shown that $\llbracket t \rrbracket_{R_\infty^\uparrow}^\xi = \llbracket \lfloor t \rfloor \rrbracket_{R_\infty}^\xi$ for all ground ceiling logic terms t . It follows that a ground equation $s \approx t$ or inequality $s \not\approx t$ is true in R_∞^\uparrow if and only if $\lfloor s \approx t \rfloor$ or $\lfloor s \not\approx t \rfloor$ is true in R_∞ . Hence, a ground clause C is true in R_∞^\uparrow if and only if $\lfloor C \rfloor$ is true in R_∞ .

By Lemma 12, R_∞ is a model of $\lfloor \mathcal{G}_\Sigma(N) \rfloor$, i.e., all clauses $\lfloor C \rfloor \in \lfloor \mathcal{G}_\Sigma(N) \rfloor$ are true in R_∞ . Hence, all clauses $C \in \mathcal{G}_\Sigma(N)$ are true in R_∞^\uparrow and therefore R_∞^\uparrow is a model of $\mathcal{G}_\Sigma(N)$.

To show that R_∞^\uparrow is term-generated, let $a \in \mathcal{U}_\tau^\uparrow$. Since $\mathcal{U}_\tau^\uparrow = \mathcal{U}_{\lfloor \tau \rfloor}$ and R_∞ is term-generated, we have a ground term t of the floor logic with $\llbracket t \rrbracket_{R_\infty}^\xi = a$. Using what we showed above, we have $\llbracket \lceil t \rceil \rrbracket_{R_\infty^\uparrow}^\xi = \llbracket t \rrbracket_{R_\infty}^\xi = a$. Hence, R_∞^\uparrow is term-generated.

Lemma 18 (Model Transfer). R_∞^\uparrow is a model of N .

Proof. We need to show that for all clauses $C \in N$, C holds in R_∞^\uparrow for all ξ . Since R_∞^\uparrow is term-generated, we have that for all variables x in C , there exists a ground term s_x such that $\xi(x) = \llbracket s_x \rrbracket_{R_\infty^\uparrow}^\xi$. Let ρ be a substitution that maps each term variable x in C to s_x and each type variable α to $\alpha\xi$. Then for any term variable x in C , $\llbracket x\rho \rrbracket_{R_\infty^\uparrow}^\xi = \xi(x)$ and for any type variable α in C , $\alpha\rho = \alpha\xi$. Then by Lemma 16 $\llbracket C \rrbracket_{R_\infty^\uparrow}^\xi = \llbracket C\rho \rrbracket_{R_\infty^\uparrow}^\xi$. As $C\rho$ is ground, by Lemma 17 it is true in R_∞^\uparrow . Thus so is C .

Lemma 19 (Extensional Model). If the extensionality axiom is present in N , then R_∞^\uparrow is an extensional model.

Proof. The proof is the same as that of Bentkamp et. al. [1]

8 Simplification Rules

A simplification inferences is an inference which renders one or more of its antecedents redundant and therefore allows them to be removed from the clause set. The efficiency of first-order superposition derives from the powerful simplification rules that can be integrated into them. Typically, a superposition prover spends far more time on simplification than on generation. First-order simplification rules can be adapted to the combinatory case quite easily by restricting attention to stable contexts.

Bentkamp et al. introduced a peculiarly higher-order simplification rule, PRUNEARG, in [2]. The rule is used to remove arguments of variables that can be represented as a function of the other arguments of the variable. They argue that the rule is sound and does not harm completeness. We reproduce the rule here and then argue the same with respect to our calculus.

$$\frac{\mathcal{C}}{(C\{y \rightarrow (\lambda \overline{x_j} . y' \overline{x_{j-1}})\}) \downarrow_w} \text{PRUNEARG}$$

where y' is a fresh variable, each occurrence of y in C has a minimum of $k \geq j$ arguments and there exists a term t such that $s_j = (t \overline{s_{j-1}} s_{j+1} \dots s_k) \downarrow_w$ for all terms $y \overline{s_k}$ occurring in C . For example, assume that $C = x a b \approx x a c$. Then PRUNEARG can be used to remove the first argument of x because there exists the term $t = \mathbf{K}$ a such that $(t b) \downarrow_w = a$ and $(t b) \downarrow_w = c$. Similarly, PRUNEARG could remove the second argument of x in the clause $C[x a a, x b b]$ where these are the only two occurrences of x in C by letting $t = \mathbf{I}$.

9 Removing Combinator Axioms

Next, we show that it is possible to replace the combinator axioms with a dedicated inference rule. We name the inference NARROW. Unlike the other inference rules, it works at prefix positions. We define *nearly first-order* positions inductively. For any term t , either $t = \zeta \overline{t_n}$ where ζ is not a fully applied combinator or $t = \tilde{\mathcal{C}} \overline{t_n}$. In the first case, the nearly first-order subterms of t are $\zeta \overline{t_i}$ for $0 \leq i \leq n$ and all the nearly first-order subterms of the t_i . In the second case, the nearly first-order subterms are $\tilde{\mathcal{C}} \overline{t_i}$ for $0 \leq i \leq n$. The notation $s \langle u \rangle$ is to be read as u is a nearly first-order subterm of s . The NARROW inference:

$$\frac{C' \vee [\neg] s \langle u \rangle \approx s'}{(C' \vee [\neg] s \langle r \rangle \approx s') \sigma} \text{NARROW}$$

with the following side conditions:

1. $u \notin \mathcal{V}$
2. Let $l \approx r$ be a combinator axiom.
 $\sigma = mgu(l, u)$;
3. $s \langle u \rangle \sigma \not\prec s' \sigma$;
4. $([\neg] s \langle u \rangle \approx s') \sigma$ is eligible in $C \sigma$,
and strictly eligible if it is positive.

We show that any inference that can be carried out using an extended combinator axiom can be simulated with NARROW proving completeness. It is obvious that an EQRES or EQFACT inference cannot have an extended combinator axiom as its premise. By the SUBVARSUP side conditions, an extended combinator axiom cannot be either of its premises. Thus we only need to show that SUP inferences with extended combinator axioms can be simulated. Note that an extended axiom can only be the left premise of a SUP inference. Consider the following inference:

$$\frac{l \approx r \quad C' \vee [\neg] s \langle u \rangle|_p \approx s'}{(C' \vee [\neg] s \langle r \rangle \approx s') \sigma} \text{SUP}$$

Let $l = \mathbf{S} \overline{x_{n>3}}$. By the variable condition, we have that $u = \zeta \overline{t_m}$ where $n \geq m \geq n-2$. If $u = y \overline{t_{n-2}}$, then $\sigma = \{y \rightarrow \mathbf{S} x_1 x_2, x_3 \rightarrow t_1, \dots, x_n \rightarrow t_{n-2}\}$. In this case $r \sigma = (x_1 x_3 (x_2 x_3) x_4 \dots x_n) \sigma = x_1 t_1 (x_2 t_1) t_2 \dots t_{n-2}$ and the conclusion of the inference is $(C' \vee [\neg] s \langle x_1 t_1 (x_2 t_1) t_2 \dots t_{n-2} \rangle \approx s') \{y \rightarrow \mathbf{S} x_1 x_2\}$. Now consider the following NARROW inference from C at the nearly first-order subterm $y t_1$:

$$\frac{C' \vee [\neg]s(\langle y t_1 \rangle t_2 \dots t_n)|_p \approx s'}{(C' \vee [\neg]s(\langle x_1 t_1 (x_2 t_1) t_2 \dots t_{n-2} \rangle \approx s'))\{y \rightarrow \mathbf{S} x_1 x_2\}} \text{NARROW}$$

As can be seen, the conclusion of the SUP inference is equivalent to that of the NARROW inference up to variable naming. The same can be shown to be the case where $u = y \overline{t_{n-1}}$ or $u = y \overline{t_n}$ or $u = \mathbf{S} \overline{t_n}$. Likewise, the same can be shown to hold when the $l \approx r$ is an extended **B**, **C**, **K** or **I** axiom.

10 Evaluation

11 Conclusion

References

1. Alexander Bentkamp, Jasmin Christian Blanchette, Simon Cruanes, and Uwe Waldmann. Superposition for lambda-free higher-order logic. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning*, pages 28–46, Cham, 2018. Springer International Publishing.
2. Alexander Bentkamp, Jasmin Christian Blanchette, Sophie Tourret, Petar Vukmirović, and Uwe Waldmann. Superposition with lambdas. *submitted for publication*, 2019.
3. François Bobot and Andrei Paskevich. Expressing polymorphic types in a many-sorted language. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *Frontiers of Combining Systems – FroCoS*, pages 87–102, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
4. Jia Meng and Lawrence C. Paulson. Translating higher-order clauses to first-order clauses. *Journal of Automated Reasoning*, 40(1):35–60, Jan 2008.
5. Stephan Schulz. E — a brainiac theorem prover. *AI Communications*, 15(2, 3):111–126, 2002.
6. Alexander Steen. *Extensional Paramodulation for Higher-Order Logic and its Effective Implementation Leo-III*. PhD thesis, Freie Universität Berlin, 2018.
7. Uwe Waldmann. Automated reasoning II. Lecture notes, Max-Planck-Institut für Informatik, 2016. <http://resources.mpi-inf.mpg.de/departments/rg1/teaching/autrea2-ss16/script-current.pdf>.