The CoRe Calculus

Serge Autexier

Saarland University & German Research Centre for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany, Email: autexier@ags.uni-sb.de

Abstract. We present the CoRe calculus for contextual reasoning which supports reasoning directly at the assertion level, where proof steps are justified in terms of applications of definitions, lemmas, theorems, or hypotheses (collectively called "assertions") and which is an established basis to generate proof presentations in natural language. The calculus comprises a uniform notion of a logical context of subformulas as well as replacement rules available in a logical context. Replacement rules operationalize assertion level proof steps and technically are generalized resolution and paramodulation rules, which in turn should suit the implementation of automatic reasoning procedures.

1 Introduction

The main application domains of computer-based theorem proving systems are mathematical assistants, mathematical teaching assistants, and hardware and software verification. In these domains, a human guidance of the proof procedures is indispensable, even for theorems that are simple by human standards. For instance the user must provide guidance information about how to explore the search space or specify intermediate lemmas. Therefore, communication between the user and the theorem proving system is crucial. The information provided by the theorem proving system about the proof must be intelligible to the user and the user must convey his/her intentions about how to continue the proof in a manner that is intelligible to the theorem proving system.

Exchanging the information in an intelligible manner is the bottleneck for the communication. A user like a mathematician or software engineer usually has a semantic representation of the problem domain and exploits it to approach and solve proof obligations. They usually have little or no knowledge about formal logic. State of the art automated theorem provers, however, only incorporate deep knowledge about the search space structure based on the syntax and calculus rules. Interactive theorem provers use tactics [11] to incorporate more high-level proof procedures, but these still stick to the syntax and the basic calculus rules. Proof planning [7] has been designed to overcome these limitations. However, in practice it also requires an understanding of the underlying calculus from the user and does not completely overcome the limitations imposed by the lack of abstraction imposed by the underlying calculus.

In this paper we present a calculus for *contextual reasoning* (CoRe) which aims at narrowing the gap between the user and the proof procedures. The

key idea of the calculus is that proof construction proceeds by transformation of (parts of) a formula by applying definitions, lemmas, theorems as well as information contained in the formula without enforcing its decomposition. We have made a point of this idea in the CoRe calculus, where the logical contexts can be statically determined for any part of a formula. The information contained in a logical context is conditioned into *replacement rules*, which formalize the notion of *assertion level* rules.

The assertion level has been introduced by Xiarong Huang in [13] as an abstraction from the pure natural deduction calculus and it is the basis for the generation of proof presentations in natural language close to the style of proofs in mathematical textbooks. The idea is to subsume axioms, definitions, lemmas, and theorems as *assertions*, and the use of a single assertion in the proof search corresponds to a whole proof segment in the underlying calculus. Consider the example assertion taken from [13]: $\forall S_1, S_2 : Set.S_1 \subseteq S_2 \Leftrightarrow \forall x.x \in S_1 \Rightarrow x \in S_2$. This assertion allows us to derive (1) $a \in S_2'$ from $a \in S_1'$ and $S_1' \subseteq S_2'$; (2) $S_1' \not\subseteq S_2'$ from $a \in S_1'$ and $a \notin S_2'$; and (3) $\forall x.x \in S_1' \Rightarrow x \in S_2'$ from $S_1' \subseteq S_2'$.

This paper is organized as follows: In Sec. 2 we recapitulate the basic definitions of higher-order logic, uniform notation, and extensional expansion proofs for higher-order logic [15]. Sec. 3 describes the CoRe calculus and how uniform notation provides a uniform basis to define the logical context of subformulas and to determine replacement rules from the assertions available from the logical context. In Sec. 4 we present an example proof with the CoRe calculus, before addressing related work and concluding in Sec. 5.

2 Preliminaries

2.1 Higher-Order Logic

For the definition of higher-order logic, we use a simple higher-order type system $\mathcal{T}$ [4], composed of a base type ι for individuals, a type o for formulas, and where $\tau \to \tau'$ denotes the type of functions from τ to τ'. As usual, we assume that the functional type constructor $\to$ associates to the right.

We annotate constants f_τ and variables x_τ with types τ from $\mathcal{T}$ to indicate their type. A higher-order signature $\Sigma = (\mathcal{T}, \mathcal{F}, \mathcal{V})$ consists of types $\mathcal{T}$, constants $\mathcal{F}$ and variables $\mathcal{V}$, both typed over $\mathcal{T}$. The typed λ-calculus is standard and is defined over a given higher-order signature $\Sigma := (\mathcal{T}, \mathcal{F}, \mathcal{V})$.

Definition 1 (λ-Terms). *Let $\Sigma = (\mathcal{T}, \mathcal{F}, \mathcal{V})$ be a higher-order signature. Then the typed λ-terms $\mathcal{T}_{\Sigma,\mathcal{V}}$ over Σ and $\mathcal{V}$ are: (Var) for all $x_\tau \in \mathcal{V}$, $x \in \mathcal{T}_{\mathcal{F},\mathcal{V}}$ is a* variable term *of type τ; (Const) for all $c_\tau \in \mathcal{F}$, $c \in \mathcal{T}_{\mathcal{F},\mathcal{V}}$ is a* constant term *of type τ; (App) if $t : \tau, t' : \tau \to \tau' \in \mathcal{T}_{\mathcal{F},\mathcal{V}}$ are typed terms, then $(t'\ t) \in \mathcal{T}_{\mathcal{F},\mathcal{V}}$ is an* application term *of type τ'; (Abs) if $x_\tau \in \mathcal{V}$ and $t : \tau' \in \mathcal{T}_{\mathcal{F},\mathcal{V}}$, then $\lambda x_\tau.t \in \mathcal{T}_{\mathcal{F},\mathcal{V}}$ is an* abstraction term *of type $\tau \to \tau'$.*

Definition 2 (Substitutions). *Let $\Sigma = (\mathcal{T}, \mathcal{F}, \mathcal{V})$ be a higher-order signature. A* substitution *is a type preserving function*[1] *$\sigma : \mathcal{V} \to \mathcal{T}_{\mathcal{F},\mathcal{V}}$ that is the identity*

[1] i.e. for all variables x_τ, $\sigma(x)$ also has type τ.

function on $\mathcal{V}$ except for finitely many elements from $\mathcal{V}$. This allows for a finite representation of a substitution σ as $\{\sigma(x_1)/x_1, \ldots, \sigma(x_n)/x_n\}$ where $\sigma(y) = y$ if $\forall 1 \leq i \leq n, y \neq x_i$.

As usual we do not distinguish between a substitution and its homomorphic extension to terms. Given a substitution σ we denote the *domain of* σ by $dom(\sigma) := \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$. Given two substitutions σ and ν, we say that ν *is a* σ*-refinement* if, and only if, there exists a substitution ρ, such that $\nu = \rho \circ \sigma$. We say that a substitution σ is *ground*, if, and only if, for all $x \in dom(\sigma)$, $\sigma(x)$ contains no free variables. Higher-order λ-terms usually come with certain reduction and expansion rules. We use the β reduction rule and the η expansion rule (see [4]), which give rise to the $\beta\eta$ long normal form, which is unique up to renaming of bound variables (α-equal). Throughout the rest of this paper we assume substitutions are idempotent[2] and all terms are in $\beta\eta$ long normal form.

For the semantics of higher-order logic we use the extensional general models from [12] by taking into account the corrections from [2]. It is based on the notion of *frames* that is a τ-indexed family $\{\overline{D}_\tau\}_{\tau \in \mathcal{T}}$ of nonempty domains, such that $\overline{D}_o = \{\top, \bot\}$ and $\overline{D}_{\tau_1 \to \tau_2}$ is a collection of functions mapping $\overline{D}_{\tau_1}$ into $\overline{D}_{\tau_2}$. Given a variable assignment ρ, a variable x_τ and an element $e \in \overline{D}_\tau$ we denote by $\rho[e/x]$ that assignment ρ' such that $\rho'(x_\tau) = e$ and $\rho'(y_{\tau'}) = \rho(y_{\tau'})$, if $y_{\tau'} \neq x_\tau$.

Definition 3 (Satisfiability & Validity). *A formula φ is satisfiable if, and only if, there is a model M and a variable assignment ρ such that $M^\rho(\varphi) = \top$. φ is* true in a model *M if, and only if, for all variable assignments ρ holds $M^\rho(\varphi) = \top$. φ is valid if, and only if, it is true in all models.*

2.2 Uniform Notation

The CoRe calculus relies on an extension of extensional expansion proofs and makes use of the concept of polarities and uniform notation (cf. [18, 9, 17]). Polarities are assigned to formulas and subformulas and are either positive ($+$) or negative ($-$). Intuitively, positive polarity of a subformula indicates that it occurs in the succedent of a sequent in a sequent calculus proof and negative polarity is for formulas occurring in the antecedent of a sequent.

Formulas annotated with polarities are called *signed formulas*. Uniform notation assigns *uniform types* to signed formulas which encode their "behavior" in a sequent calculus proof: there are two propositional uniform types α and β, and two types γ and δ for quantification over object variables. A signed formula is of type α if the subformulas obtained by application of the respective sequent calculus decomposition rule on the formula both occur in the same sequent. Signed formulas are of type β, if the decomposition of the signed formula gives rise to a split of the sequent calculus proof and the obtained subformulas occur in different sequents. γ-type signed formulas indicate that the bound variable is freely instantiable, while δ-type signed formulas are those for which the *Eigenvariable*

[2] i.e. for all terms t holds $\sigma(\sigma(t)) = \sigma(t)$.

α	α_0	α_1
$(\varphi \vee \psi)^+$	φ^+	ψ^+
$(\varphi \Rightarrow \psi)^+$	φ^-	ψ^+
$(\varphi \wedge \psi)^-$	φ^-	ψ^-
$(\neg\varphi)^+$	φ^-	$-$
$(\neg\varphi)^-$	φ^+	$-$

β	β_0	β_1
$(\varphi \wedge \psi)^+$	φ^+	ψ^+
$(\varphi \vee \psi)^-$	φ^-	ψ^-
$(\varphi \Rightarrow \psi)^-$	φ^+	ψ^-

γ	$\gamma_0(c)$
$(\forall x.\varphi)^-$	$(\varphi[x/t])^-$
$(\exists x.\varphi)^+$	$(\varphi[x/t])^+$

δ	$\delta_0(c)$
$(\forall x.\varphi)^+$	$(\varphi[x/c])^+$
$(\exists x.\varphi)^-$	$(\varphi[x/c])^-$

ϵ	ϵ_0	ϵ_1
$(s \Leftrightarrow t)^-$	s	t
$(s = t)^-$	s	t

ζ	ζ_0	ζ_1
$(s \Leftrightarrow t)^+$	s	t
$(s = t)^+$	s	t

Fig. 1. Extended Uniform Notation.

condition must hold. We call γ-*variable* (resp. δ-*variable*) variables bound on some γ-type signed formula (resp. δ-type). In Fig. 1 we give the list of signed formulas for each uniform type.

An important intuitive concept is equality and equivalence and we want to treat those as first-class citizens by supporting their use as rewrite rules. For instance, given an equation $s = t$ and a formula $\varphi(s)$ it is natural to allow the rewrite of $\varphi(s)$ to $\varphi(t)$. Similarly we want to support the rewriting with equivalence, i.e. to apply $P \Leftrightarrow Q$ on $\varphi(P)$ to obtain $\varphi(Q)$. Note that we cannot assign polarities to P and Q in $P \Leftrightarrow Q$, while P in $\varphi(P)$ may well have a polarity. Furthermore, the uniform notion of rules obtained from uniform notation is restricted to logical refinement rules and does not capture equivalence rules. In order to capture equations and equivalences we introduce two new uniform types ϵ and ζ respectively for negative and positive equations and equivalences (see the rightmost tables in Fig. 1).

In the following we agree to denote by $\alpha^p(F^q, G^r)$ signed formulas of polarity p, uniform type α, and subformulas F and G with respective polarities q and r according the tables in Fig. 1, including the unary version $\alpha^p(F^q)$. Note that the subformulas are not necessarily direct subformulas, as for instance $\alpha^+(F^-, G^+)$ denotes $(F \Rightarrow G)^+$ but also $(\neg F \vee G)^+$. By abuse of notation we also allow the replacement of F^q and G^r by new formulas. Example: if $\alpha^+(F^q, G^r)$ is $(A^- \Rightarrow B^+)^+$, then $\alpha^p(C, G^r)$ denotes $(C^- \Rightarrow B^+)^+$. We use an analogous notation for formulas of the other uniform types. Furthermore we define $\overline{\alpha}^p(F^q) := \alpha^p(F^q)$ and for $n > 1$, $\overline{\alpha}^p(F_1^{p_1}, \ldots, F_n^{p_n}) := \alpha^p(F_1^{p_1}, \overline{\alpha}^{p_2}(F_2^{p_2}, \ldots, F_n^{p_n}))$. Analogously we define $\overline{\beta}$, but also add the case $n = 0$ by defining $\overline{\beta}^+() := \top^+$ and $\overline{\beta}^-() := \bot^-$.

In the rest of this article we are mainly concerned with signed formulas. To ease the presentation we extend the notion of satisfiability to signed formulas. In order to motivate this definition consider a sequent $\psi_1, \ldots, \psi_n \vdash \varphi$. It represents the proof status that we have to prove φ from the ψ_i. In terms of polarities, all the ψ_i have negative polarity while φ has positive polarity. The ψ_i are the assumptions and thus we consider the models that satisfy those formulas and prove that those models also satisfy φ. Hence, we define that a model M satisfies a *negative* formula ψ_i^- if, and only if, M satisfies ψ_i. From there we derive the dual definition for positive formulas, namely that a model M satisfies a positive formula F^+, if, and only if, M *does not* satisfy F.

Definition 4 (Satisfiability of Signed Formulas). *Let F^p be a signed formula of polarity p, M a model and ρ an assignment. Then we define: $M^\rho \models F^+$* holds if, and only if, *$M^\rho \not\models F$. Conversely, we define $M^\rho \models F^-$* holds if, and only

if, $M^\rho \models F$. *We extend that notion to sets of signed formulas* $\mathcal{F}$ *by:* $M^\rho \models \mathcal{F}$, *if, and only if, for all* $F^p \in \mathcal{F}$, $M^\rho \models F^p$.

Lemma 1. *Let* M *be a model,* ρ *a variable assignment,* F^q, G^r *signed formulas of polarities* q, r. *Then* $M^\rho \models \alpha^p(F^q, G^r)$ *holds if, and only if, both* $M^\rho \models F^q$ *and* $M^\rho \models G^r$ *hold;* $M^\rho \models \beta^p(F^q, G^r)$ *holds, if, and only if,* $M^\rho \models F^q$ *or* $M^\rho \models G^r$ *holds.*

Remark 1 (Notational Conventions). We denote formulas by capital Latin letters $A, B, \ldots, F, G, \ldots$, and formulas with holes by Greek letters $\varphi(.)$, which denotes λ-abstractions $\lambda x.\varphi(x)$ where x occurs *exactly* once in φ. Similarly, we define $\psi(.,.)$ to denote $\lambda x.\lambda y.\psi(x,y)$ and x and y occur exactly once in ψ. [3]

Definition 5 (Literals in Signed Formulas). *Let* $\varphi(F)^p$ *be a signed formula of polarity* p. *We say that* F *is a* literal *in* $\varphi(F)^p$ *if using the rules from Fig. 1 we can assign a polarity to* F, *but not to its subterms.*

2.3 Extensional Expansion Proofs

For the CoRe calculus we build upon an extension of extensional expansion proofs from [3]. They rely on extensional expansion trees which are defined in [15] for formulas without equivalences, equations and the atoms $\top$ and $\bot$ and where all negations are around the literals. They are constructed for a formula F along the tree structure of the formula. It follows closely that tree structure, except for existential quantifiers, where it allows to have subtrees for arbitrary many instances t of the quantifier. Each t is called an *expansion term* and the number of instances is the multiplicity of that quantifier. For a universal quantifier the subtree is created for the main formula instantiated with a *selected parameter*. The non-instantiated formula represented by a subtree is called the *shallow formula*. To each subtree we can associate a *deep formula* which represents the formula where all quantifiers have been instantiated with expansion terms. Equations are handled via Leibniz' definition of equality in [15]. It also adds a rule to support reasoning with respect to functional and Boolean extensionality, which gives rise to *extensional expansion trees*. We denote the extensional expansion trees with shallow formula F and multiplicity μ by $\triangle_F^\mu$. The expansion terms in an expansion tree give rise to a substitution, which is $\triangle_F^\mu$*-admissible*, if the transitive closure of the relation induced by the hierarchy of quantifiers from the tree structure together with the relation induced by the substitution among introduction nodes of expansion terms and selected parameters is irreflexive. An extensional expansion tree $\triangle_F^\mu$ is an extensional expansion proof for F, if, and only if, (1) its deep formula is a tautology (i.e., all paths through the deep formula are unsatisfiable) and (2) the associated substitution σ is $\triangle_F^\mu$-admissible.

In [3] we extend that calculus (1) by allowing for equivalences, equations and the atoms $\top$ and $\bot$; (2) by using polarities to overcome the restrictions on the position of negations[4]; (3) by including a rule to dynamically increase the

[3] This is similar to the notation used by Schütte in [16].

[4] Here we follow the *indexed formula trees* of Wallen [18] for first-order modal logics and fragments thereof.

multiplicities on the fly in order to avoid guessing the initial multiplicities and restart. Increasing the multiplicities thereby copies relevant existing connections and results in a renaming of existential (γ) variables and universal (δ) variables.

Theorem 1 (Soundness & Completeness with Dynamic Increase of Multiplicities). *F is valid if, and only if, from an extensional expansion tree for the signed formula F^+ with singular multiplicities we can derive an extensional expansion tree and a $\triangle_F^\mu$-admissible substitution σ such that all paths in $\triangle_F^\mu$ are unsatisfiable.*

3 CoRe Calculus

The CoRe calculus relies on the idea to have a single signed formula representing the state of the proof and to use polarities and uniform types to manipulate the subformulas. In order to check the admissibility of substitutions it uses the extensional expansion trees from [3] (see previous section). For a closed signed formula F^p it constructs an extensional expansion tree with singular multiplicities and with γ- and δ-variables. From such an initial extensional expansion tree, we take its deep formula with free γ- and δ-variables.

As an example consider the closed formula representing the structural induction axiom $\forall p_{\iota\to o}.(\forall n_\iota.n = 0 \Rightarrow p(n)) \Rightarrow ((\forall n, n'_\iota.(n = n'+1 \wedge p(n')) \Rightarrow p(n)) \Rightarrow \forall n_\iota.p(n))$. Writing free γ-variables in capital letters, δ-variables in lower-case letters and performing the necessary renamings to avoid name clashes, the free variable representation of the extensional expansion tree for the negative version of the formula and with singular multiplicities is $((n_1 = 0 \Rightarrow P(n_1)) \Rightarrow (((n_2 = n_3 + 1 \wedge P(n_3)) \Rightarrow P(n_2)) \Rightarrow P(N)))^-$. Note that the free γ- and δ-variables are bound in the corresponding extensional expansion tree. The extensional expansion tree this formula stems from can be used to check the admissibility of any substitution for this formula, even if the formula is further transformed.

The obtained signed formulas have the property that they do not contain any quantifier with defined polarity, but may well contain quantifiers inside literals. We denote this kind of signed formulas as *quantifier-free formulas.*

Definition 6 (Quantifier-Free & Ground Signed Formulas). *Let F^p be a signed formula of polarity P. We say that F^p* is quantifier free (QF)*, if F^p does not contain any signed subformula of uniform type δ or γ. If F^p additionally contains no free γ-variables, then F^p* is ground.

An example for a ground signed formula which contains quantifiers inside literals is the following definition of the predicate of even natural numbers $(\mathsf{Even}_{\iota\to o} = \lambda x_\iota.\exists n_\iota.x = 2 \times n)^p$. Given a closed formula F, we denote by $\triangle_F^1$ the initial extensional expansion tree of singular multiplicity for F and by F_X^+ the quantifier free formula obtained from $\triangle_F^1$. Using polarities and uniform types we can define relationships between subformulas in signed QF-formulas as well as the *set of set of paths* through signed quantifier-free formulas.

Definition 7 (α- and β-Related Signed Formulas). *Let $\psi(F,G)^p$ be a signed formula of polarity p. We say that F* and *G* are α-related (resp. β-related) in *$\psi(F,G)^p$ if, and only if, the smallest signed subformula of $\psi(F,G)^p$ which contains both F and G is of uniform type α (resp. β).*

Definition 8 (Paths in Signed QF-Formula). *Let F^p be a signed quantifier-free formula of polarity p. A path in F^p is a sequence $\ll F_1^{p_1}, \ldots, F_n^{p_n} \gg$ of α-related signed subformulas of F^p. The sets $\mathcal{P}(F^p)$ of paths through F^p is the smallest set containing $\{\ll F^p \gg\}$ and which is closed under the two operations:*

(α) *If $P \cup \{\ll \Gamma, \alpha^p(G^q, H^r) \gg\} \in \mathcal{P}(F^p)$, then $P \cup \{\ll \Gamma, G^q, H^r \gg\} \in \mathcal{P}(F^p)$.*
(β) *If $P \cup \{\ll \Gamma, \beta^p(G^q, H^r) \gg\} \in \mathcal{P}(F^p)$, then $P \cup \{\ll \Gamma, G^q \gg, \ll \Gamma, H^r \gg\} \in \mathcal{P}(F^p)$.*

The following lemma establishes the basic relationship between the paths in the extensional expansion tree for a closed formula and the paths through the corresponding quantifier-free signed formula.

Lemma 2. *Let F be a closed formula, $\triangle_F^1$ the extensional expansion tree with singular multiplicity for F^+ and F_X^+ the QF-formula for $\triangle_F^1$. Then the set of all literal paths through $\triangle_F^1$ are in $\mathcal{P}(F_X^+)$.*

This allows us to derive from the existence of satisfiable (ground) literal paths in the extensional expansion tree the existence of (ground) literal paths in the QF-formula. The preservation of the existence of satisfiable ground paths is the property ensuring the soundness of any further transformation performed on the QF-formula. Since it is cumbersome to always have to reason about the existence of ground paths *only* at the level of literals, we lift the level to paths at an arbitrary, less granular level of paths.

Definition 9 (Satisfiable & Unsatisfiable Ground Paths). *A path p is* ground *if all contained signed formulas are ground. A ground path p* is satisfiable *if there exists a model M such that $M \models p$. Otherwise p is* unsatisfiable.

Corollary 1. *Let p be a path, which contains either $\top^+$, or $\bot^-$ or signed formulas F^- and F^+. Then any ground path for p is unsatisfiable.*

The following lemma establishes then that we can choose the level of granularity on which we want to reason about the satisfiability of a complete set of paths.

Lemma 3. *Let F^p be a signed formula, and $P, P' \in \mathcal{P}(F^p)$ complete sets of ground paths through F^p. Then it holds: P contains a satisfiable ground path if, and only if, P' contains a satisfiable ground path.*

3.1 Admissible Replacement Rules

The logical context of some subformula in a signed formula is determined by collecting all α-related subformulas. Now we are concerned with determining

the possible rules which can be generated from the available subformulas. To motivate this, consider the goal sequent $A \Rightarrow (B \Rightarrow C) \vdash C$. Applying $A \Rightarrow (B \Rightarrow C)$ to C means that the goal to prove C is replaced by the goal to prove A and B. In this case both occurrences of C have opposite polarities and are α-related via $\vdash$. Furthermore, the new subgoals, i.e. the positive occurrences of A and B, can be determined statically from the formula by collecting all the formulas that are β-related to the negative occurrence of C. This enables generating rules from a formula by fixing the left-hand side, e.g. the negative C. The right-hand side of the rule is then the list of all formulas that are β-related to the left-hand side, and we write this as a rule $C^- \rightarrow \langle A^+, B^+ \rangle$. Analogously, if there is a negative equation or a negative equivalence in the context, i.e. an ϵ-type formula $\epsilon(s,t)$, we obtain the rules $s \rightarrow \langle t, F_1^{p_1}, \ldots, F_n^{p_n} \rangle$ and $t \rightarrow \langle s, F_1^{p_1}, \ldots, F_n^{p_n} \rangle$ where the $F_i^{p_i}$ are the formulas β-related to $\epsilon(s,t)$. This rule contains the information, that some goal formula $\varphi(s)$ where s has an arbitrary polarity – even no polarity – can be refined to the subgoals $\varphi(t), F_1^{p_1}, \ldots, F_n^{p_n}$.

Before formalizing the notion of replacement rules, we introduce a mechanism which allows to weaken parts of α-type signed subformulas.

Definition 10 (Weakening of Signed Formulas). *Let F^p be a signed formula. The set $\mathcal{W}(F^p)$ of weakened signed formulas for F^p is defined recursively over the structure of F: (Atom) $\mathcal{W}(F^p) := \{F^p\}$ if F^p is a literal; (α) $\mathcal{W}(\alpha^p(G^q, H^r)) := \{\alpha^p(G_w^q, H_w^r) \mid G_w^q \in \mathcal{W}(G^q), H_w^r \in \mathcal{W}(H^r)\} \cup \mathcal{W}(G^q) \cup \mathcal{W}(H^r)$; ($\beta$) $\mathcal{W}(\beta^p(G^q, H^r)) := \{\beta^p(G_w^q, H_w^r) \mid G_w^q \in \mathcal{W}(G^q), H_w^r \in \mathcal{W}(H^r)\}$.*

Lemma 4. *Let M be a model, F^p a signed formula and σ a ground substitution such that $\sigma(F)^p$ is ground. Then for any $G^p \in \mathcal{W}(F^p)$ it holds: If $M \models \sigma(F^p)$ then $M \models \sigma(G^p)$.*

Proof. By structural induction over F and using Lemma 1. □

Corollary 2 (Connectable Signed Formulas). *Let F^p and G^{-p} be two signed formulas. If there exists an $F_w^p \in \mathcal{W}(F^p)$ which is α-equal to some $F_w^{-p} \in \mathcal{W}(G^{-p})$, then for any ground substitution σ such that $\sigma(F^p)$ and $\sigma(G^p)$ are ground there exists no model M which satisfies both $\sigma(F^p)$ and $\sigma(G^{-p})$. We say that F^p and G^{-p} are* connectable.

Example 1. As an example consider the negative formula $A \vee (B \wedge C)^-$ and the positive formula $A \vee (C \vee D)^+$. The respective sets of weakened signed formulas are $\mathcal{W}(A \vee (B \wedge C)^-) = \{A \vee (B \wedge C)^-, A \vee B^-, A \vee C^-\}$ and $\mathcal{W}(A \vee (C \vee D)^+) = \{A \vee (C \vee D)^+, A \vee C^+, A \vee D^+, A^+, C^+, D^+\}$. Thus, the formulas are connectable, since $A \vee C^- \in \mathcal{W}(A \vee (B \wedge C)^-)$ and $A \vee C^+ \in \mathcal{W}(A \vee (C \vee D)^+)$.

Definition 11 (Subformula Conditions). *Let $\varphi(F^q)^p$ be a signed formula of polarity p, $G_1^{p_1}, \ldots, G_n^{p_n}$ be all maximal signed formulas that are β-related to F^q in $\varphi(F^q)^p$. Then the conditions of F^q are $\mathcal{C}_{\varphi(.)^p} := \mathcal{W}(G_1^{p_1}) \times \ldots \times \mathcal{W}(G_n^{p_n})$.*

Replacement rules are of two kinds: the first kind are those where the left-hand side is a subformula with a polarity, and the second kind result from ϵ-type formulas. The former are called *resolution replacement rules*, while the latter are called *rewriting replacement rules*.

Definition 12 (Admissible Resolution Replacement Rules). *Given the signed formula $\psi(L^{-p}, G^p)^q$ of polarity q and containing the two signed subformulas L and G of opposite polarities, let $\psi'(L^{-p}, G^p)^r$ be the smallest signed formula containing both L^{-p} and G^p and $(R_1^{p_1}, \ldots, R_n^{p_1}) \in \mathcal{C}_{\psi'(.,G^p)^r}$. Then if L^{-p} and G^p are α-related in $\psi(L^{-p}, G^p)^q$, then $L^{-p} \to \langle R_1^{p_1}, \ldots, R_n^{p_n} \rangle$ is an* admissible resolution replacement rule *for G^p.*

Definition 13 (Admissible Rewriting Replacement Rules). *Given the signed formula $\psi(\epsilon^-(s,t), G^p)^q$ of polarity q and containing the two signed subformulas $\epsilon^-(s,t)$ and G^p, let $\psi'(\epsilon^-(s,t), G^p)^r$ be the smallest signed formula containing both $\epsilon^-(s,t)$ and G^p and $(R_1^{p_1}, \ldots, R_n^{p_1}) \in \mathcal{C}_{\psi'(.,G^p)^r}$. Then if $\epsilon^-(s,t)$ and G^p are α-related in $\psi(\epsilon^-(s,t), G^p)^q$, then $s \to \langle t, R_1^{p_1}, \ldots, R_n^{p_n} \rangle$ and $t \to \langle s, R_1^{p_1}, \ldots, R_n^{p_n} \rangle$ are* admissible rewriting replacement rules *for G^p.*

A CoRe proof state is denoted by $\triangle; \sigma \triangleright F$ and consists of an extensional expansion tree $\triangle$, a $\triangle$-admissible substitution σ and a QF-formula F. We say that $\triangle; \sigma \triangleright F$ *is satisfiable* if, and only if, for all $\triangle$-admissible ground σ-refinements ν there exists a satisfiable path in $\nu(F)^+$. In order to mimic a textual top-down proof development style, the CoRe calculus rules $\frac{\pi}{\pi'}$ are to be read top-down[5], i.e. the problem of proving π is reduced to prove π'. The rules are given in Fig. 2 and consist of three parts separated by dotted lines: The upper part presents the major calculus rules, the middle part consists of the propositional simplification rules, and the lower part gives an extra rule for the application of rewriting replacement rule. The rules from the first two parts are the kernel of the CoRe calculus necessary for completeness. The last rule is added for convenience, but is admissible. We only illustrate some of the rules in detail and the meaning of the other rules should follow easily from these descriptions.

The $Weak_L$ rule allows to reduce the goal to prove a positive formula φ in which occurs a binary α-type subformula $\alpha^p(F^q, G^r)$ to the goal to prove the formula φ, where the binary subformula is replaced by the left conjunct F^q. The surrounding $\overline{\alpha}^p$ is used to adapt the polarities by adding a negation, in case $p \neq q$. Example applications of this rule are: $\triangle; \sigma \triangleright C \wedge (A \Rightarrow B) \vdash_C^{Weak_L} \triangle; \sigma \triangleright C \wedge B$, or $\triangle; \sigma \triangleright C \wedge (A \Rightarrow B) \vdash_C^{Weak_L} \triangle; \sigma \triangleright C \wedge \neg(A)$. This rule changes neither the extensional expansion tree nor the substitution.

The *Leibniz*-rule is used to expand an equality $s =_\tau t$ into Leibniz definition of equality $\forall P_{\tau \to o}. P(s) \Rightarrow P(t)$, for any type τ. Therefore we determine the extensional expansion subtree $\triangle_{s=t^p}$ for $(s=t)^p$ and apply the respective Leibniz-Introduction-rule of extensional expansion proofs. This adds an initial extensional expansion subtree $\triangle_{\forall P.\, P(s) \Rightarrow P(t)}$ conjunctively to $\triangle_{s=t^p}$. From that new subtree we take the corresponding QF-formula $P(s) \Rightarrow P(t)$ and α-relate it to $s = t^p$ to obtain the new QF-formula $\varphi(\alpha^p(s = t^p, P(s) \Rightarrow P(t)))$. Depending on whether the polarity p is positive or not, P is a new δ-variable or a new γ-variable. An example for this rule is: $\triangle_{\triangle_{\mathrm{s}(X)+Y=\mathrm{s}(X+Y)^-}}; \sigma \triangleright (X'+0 = X' \wedge \mathrm{s}(X)+Y = \mathrm{s}(X+Y)) \Rightarrow \mathrm{s}(\mathrm{s}(a))+0 = \mathrm{s}(\mathrm{s}(a)) \vdash_C^{Leibniz} \triangle_{\alpha^-(\triangle_{\mathrm{s}(X)+Y=\mathrm{s}(X+Y)}, \triangle_{\forall P.\, P(\mathrm{s}(X)+Y) \Rightarrow P(\mathrm{s}(X+Y)))})}; \sigma \triangleright$

[5] Unlike sequent calculus rules.

$$\frac{\triangle;\sigma \rhd \top}{q.e.d.}\ \textit{Axiom} \qquad \frac{\triangle;\sigma \rhd \varphi(F^p)}{\triangle;\sigma \rhd \varphi(\alpha^p(F^p,F^p))}\ \textit{Contract} \qquad \frac{\triangle;\sigma \rhd \varphi((F \Leftrightarrow G)^+)}{\triangle;\sigma \rhd \varphi(((F \Rightarrow G) \wedge (G \Rightarrow F))^+)}\ \zeta\textit{-Elim}$$

$$\frac{\triangle;\sigma \rhd \varphi(\alpha^p(F^q,G^r))}{\triangle;\sigma \rhd \varphi(\overline{\alpha}^p(F^q))}\ \textit{Weak}_L \qquad \frac{\triangle;\sigma \rhd \varphi(\alpha^p(F^q,G^r))}{\triangle;\sigma \rhd \varphi(\overline{\alpha}^p(G^r))}\ \textit{Weak}_R$$

$$\frac{\triangle_{\triangle_{s=t^p};\sigma \rhd \varphi(s=t^p)}}{\triangle_{\alpha^p(\triangle_{s=t^p},\triangle_{\forall P.\,P(s)\Rightarrow P(t)});\sigma \rhd \varphi(\alpha^p(s=t,P(s)\Rightarrow P(t)))}}\ \textit{Leibniz}$$

$$\frac{\triangle_{s=t^p};\sigma \rhd \varphi(s=t^p)}{\triangle_{\alpha^p(s=t^p,\lambda x.\,s=\lambda x.\,t);\sigma \rhd \varphi(\alpha^p(s=t,\lambda x.s=\lambda x.t))}}\ \textit{f-Ext}$$

if x local to $s=t$ in $\varphi(s=t^p)$.

$$\frac{\triangle_{A\Leftrightarrow B^p};\sigma \rhd \varphi(A \Leftrightarrow B^p)}{\triangle_{\alpha^p(A\Leftrightarrow B^p,\lambda x.\,A=\lambda x.\,B);\sigma \rhd \varphi(\alpha^p(A\Leftrightarrow B,\lambda x.A=\lambda x.B))}}\ \textit{b-Ext}$$

if x local to $A \Leftrightarrow B$ in $\varphi(A \Leftrightarrow B^p)$.

$$\frac{\triangle;\sigma \rhd F}{\sigma'(\triangle);\sigma' \circ \sigma \rhd \sigma'(F)}\ \textit{Subst} \qquad \frac{\triangle;\sigma \rhd \varphi(F^p)}{\triangle;\sigma \rhd \varphi(\overline{\beta}^p(V_1^{p_1},\ldots,V_n^{p_n}))}\ \textit{Res}$$

if $\sigma' \circ \sigma$ admissible wrt. $\triangle$.

If $U^{-p} \to \langle V_1^{p_1},\ldots,V_n^{p_n}\rangle$ admissible for F^p and U^{-p} and F^p are connectable.

$$\frac{\triangle_{\triangle_1\cdots\triangle_n};\sigma \rhd \varphi(\psi_1(\overrightarrow{V_1}),\ldots,\psi_n(\overrightarrow{V_k}))}{\triangle_{\triangle_1\triangle'_1\cdots\triangle_n,\triangle'_n};\sigma' \circ \sigma \rhd \varphi(\alpha(\psi_1(\overrightarrow{V_1}),\psi_1(\rho(\overrightarrow{V_1}))),\ldots,\alpha(\psi_n(\overrightarrow{V_k}),\psi_n(\rho(\overrightarrow{V_k}))))}\ \mu\textit{-Inc}$$

where ρ is the renaming of γ- and δ-variables declared in the $\triangle_i$, $(\overrightarrow{V_i})_{i=1\ldots k}$ is a disjoint partition of $dom(\rho)$, such that for all i, all the variables in $\overrightarrow{V_i}$ occur only in $\psi_i(\overrightarrow{V_i})$, and $\sigma' := [\rho(\sigma(x))/\rho(x) \mid \text{for all } \gamma\text{-variables } x \in dom(\rho)]$.

$$\frac{\triangle_{\triangle_{F^p};\sigma \rhd \varphi(F^p)}}{\triangle_{\triangle_C};\rho \circ \sigma \rhd \varphi(\beta^p(\alpha^p(A^-,F^p),\alpha^p(A^+,F^p)))}\ \textit{Cut } A$$

where $\triangle_{F^p}$ is the minimal subtree containing all literals in F^p, $\overrightarrow{x}'$ are the free variables of A not bound above $\triangle_{F^p}$, $\rho := [\overrightarrow{x}'/\overrightarrow{x}]$, and $\triangle_C$ is the subtree $\gamma^p\overrightarrow{x}'.\beta^p(\alpha^p(A^-,\triangle_{F^p}),\alpha^p(A^+,\triangle_{F^p}))$.

$$\frac{\triangle;\sigma \rhd \varphi(\top \vee F)}{\triangle;\sigma \rhd \varphi(\top)}\ \vee_L^\top \quad \frac{\triangle;\sigma \rhd \varphi(F \vee \top)}{\triangle;\sigma \rhd \varphi(\top)}\ \vee_R^\top \quad \frac{\triangle;\sigma \rhd \varphi(\bot \vee F)}{\triangle;\sigma \rhd \varphi(F)}\ \vee_L^\bot \quad \frac{\triangle;\sigma \rhd \varphi(F \vee \bot)}{\triangle;\sigma \rhd \varphi(F)}\ \vee_R^\bot$$

$$\frac{\triangle;\sigma \rhd \varphi(\top \Rightarrow F)}{\triangle;\sigma \rhd \varphi(F)}\ \Rightarrow_L^\top \quad \frac{\triangle;\sigma \rhd \varphi(F \Rightarrow \top)}{\triangle;\sigma \rhd \varphi(\top)}\ \Rightarrow_R^\top \quad \frac{\triangle;\sigma \rhd \varphi(\bot \Rightarrow F)}{\triangle;\sigma \rhd \varphi(\top)}\ \Rightarrow_L^\bot \quad \frac{\triangle;\sigma \rhd \varphi(F \Rightarrow \bot)}{\triangle;\sigma \rhd \varphi(\neg(F))}\ \Rightarrow_R^\bot$$

$$\frac{\triangle;\sigma \rhd \varphi(\top \wedge F)}{\triangle;\sigma \rhd \varphi(F)}\ \wedge_L^\top \quad \frac{\triangle;\sigma \rhd \varphi(F \wedge \top)}{\triangle;\sigma \rhd \varphi(F)}\ \wedge_R^\top \quad \frac{\triangle;\sigma \rhd \varphi(\bot \wedge F)}{\triangle;\sigma \rhd \varphi(\bot)}\ \wedge_L^\bot \quad \frac{\triangle;\sigma \rhd \varphi(F \wedge \bot)}{\triangle;\sigma \rhd \varphi(\bot)}\ \wedge_R^\bot$$

$$\frac{\triangle_{\epsilon(s,t)};\sigma \rhd \varphi(L(s)^p)}{\triangle_{\alpha^-(\epsilon(s,t),\forall P.\,\beta^-(L(s)^{-p},L(t)^p))};[\lambda x.L(x)/P] \circ \sigma \rhd \varphi(\overline{\beta}^p(L(t)^p,V_1^{p_1},\ldots,V_n^{p_n}))}\ \textit{Rew}$$

If $L(s)^p$ is a literal, $s \to \langle t, V_1^{p_1},\ldots,V_n^{p_n}\rangle$ is admissible for $L(s)^p$, and $\epsilon(s,t)$ is the equation or equivalence this rule results from.

Fig. 2. The CoRe Calculus

$$\begin{array}{ccc} x & \xrightarrow{\rho} & \rho(x) = x' \\ {\scriptstyle\sigma}\downarrow & & \downarrow{\scriptstyle\sigma'} \\ \sigma(x) & \xrightarrow{\rho} & \sigma'(x') := \rho(\sigma(x)) \end{array}$$

(a) Construction of σ'

$$\begin{array}{l} \triangle_{\alpha(\triangle_F, \triangle_{F'})}; \sigma' \circ \sigma \triangleright \\ \left(\begin{array}{l} X' + 0 = X' \wedge (\mathrm{s}(X) + Y = \mathrm{s}(X + Y)) \\ \wedge(\mathrm{s}(\mathrm{s}(X) + Y) + 0 = \mathrm{s}(\mathrm{s}(X) + Y) \\ \quad \Rightarrow \mathrm{s}(\mathrm{s}(X + Y)) + 0 = \mathrm{s}(\mathrm{s}(X + Y)))) \\ \wedge(\mathrm{s}(U) + V = \mathrm{s}(U + V)) \\ \wedge(\mathrm{s}(\mathrm{s}(U) + V) + 0 = \mathrm{s}(\mathrm{s}(U) + V) \\ \quad \Rightarrow \mathrm{s}(\mathrm{s}(U + V)) + 0 = \mathrm{s}(\mathrm{s}(U + V)))) \end{array} \right) \\ \Rightarrow \mathrm{s}(\mathrm{s}(a)) + 0 = \mathrm{s}(\mathrm{s}(a)) \end{array}$$

(b) Proof state after increase of multiplicity

Fig. 3. (a) Construction of σ' (b) Proof state after increase of multiplicity

$(X'+0 = X' \wedge (\mathrm{s}(X)+Y = \mathrm{s}(X+Y) \wedge (P(\mathrm{s}(X)+Y) \Rightarrow P(\mathrm{s}(X+Y))))) \Rightarrow \mathrm{s}(\mathrm{s}(a))+0 = \mathrm{s}(\mathrm{s}(a))$, where P is a new γ-variable.

The functional and Boolean extensionality rules *f-Ext* and *b-Ext* require the variable x to be local with respect to the equation (resp. equivalence). This intuitively means that if x is a γ-variable, then it does not occur in a part which is β-related to the equation (resp. equivalence). If x is a δ-variable, then it does not occur in a part which is α-related. For a formalization we refer to [3].

The rule *Res* is the central rule operationalizing assertion level reasoning. Assertions G can be applied to some subformula F^p if it is possible to compile from G an admissible resolution replacement rule $U^{-p} \to \langle V_1^{p_1}, \ldots, V_n^{p_n} \rangle$. Such a rule is applicable, if F^p and U^{-p} are connectable, and the application consists of replacing F^p by the disjunction of the subgoals $V_1^{p_1}, \ldots, V_n^{p_n}$, which is represented by $\overline{\beta}^p(V_1^{p_1}, \ldots, V_n^{p_n})$. Examples for this rule are: $\triangle; \sigma \triangleright (A \Rightarrow B) \Rightarrow B \vdash_C^{Res} \triangle; \sigma \triangleright (A \Rightarrow B) \Rightarrow A$ by the rule $B^- \to \langle A^+ \rangle$, $\triangle; \sigma \triangleright (A \vee (B \wedge C)) \Rightarrow B \vdash_C^{Res} \triangle; \sigma \triangleright (A \vee (B \wedge C)) \Rightarrow \neg(A)$ by the rule $B^- \to \langle A^- \rangle$, or $\triangle; \sigma \triangleright (A \Rightarrow B) \Rightarrow B \vdash_C^{Res} \triangle; \sigma \triangleright (A \Rightarrow \bot) \Rightarrow B$ by the rule $B^+ \to \langle . \rangle$ where by definition $\overline{\beta}^-(.) = \bot^-$.

Finally, the μ-*Inc*-rule allows to increase the multiplicities of γ-quantifiers in the extensional expansion tree. This rule is essentially necessary to be combined with the substitution rule in order to preserve formulas containing those variables that will be substituted. Assume $\{x_1, \ldots, x_n\}$ are those γ-variables that will be substituted and whose associated subtrees $\triangle_i$ are maximal with respect to any other substituted γ-variable. The $\triangle_i$ are the immediate subtrees in $\triangle$ of the γ-quantifier that introduced x_i. The increase of the multiplicity of these γ-quantifiers with respect to the x_i results in copies $\triangle_i'$ of the $\triangle_i$, where all γ- and δ-variables declared in $\triangle_i$ have been renamed. This renaming ρ is injective and maps γ-variables to γ-variables and δ-variables to δ-variables. Furthermore, it holds $\rho(x_i) = x_i'$ for all $1 \leq i \leq n$. The extension σ' of the substitution σ is obtained by making the diagram on the left hand side of Fig. 3 commute for any γ-variable x in the domain of ρ. Finally, the renaming is propagated to the QF-formula by considering a partitioning $(\overrightarrow{V_i})_{i=1\ldots k}$ of the renamed γ- and δ-variables ($dom(\rho)$), such that for all $1 \leq i \leq k$ there is a subformula $\psi_i(\overrightarrow{V_i})$ containing *all* occurrences of the variables in $\overrightarrow{V_i}$. The renaming ρ is applied to these $\psi_i(\overrightarrow{V_i})$ to obtain $\psi_i(\rho(\overrightarrow{V_i}))$, which in turn are added conjunctively to $\psi_i(\overrightarrow{V_i})$. Consider the proof state obtained before by the *Leibniz*-rule: We first apply the substitution $\sigma(P) = \lambda x.\mathrm{s}(x + 0) = \mathrm{s}(x)$ and then increase the multiplicity of

$$\forall p.\forall x.\exists y.(p(0) \wedge (p(y) \Rightarrow p(\mathrm{s}(y)))) \Rightarrow p(x) \quad (1)$$
$$\forall n.\sum_{i=1}^{0} i^n = 0 \quad (2)$$
$$\forall n, m.\sum_{i=1}^{\mathrm{s}(m)} i^n = \mathrm{s}(m)^n + \sum_{i=1}^{m} i^n \quad (3)$$
$$\forall a, b.(a+b)^2 = (a^2 + (2 \times b) \times a) + b^2 \quad (4)$$
$$\forall a, b, c.a = b \Rightarrow a + c = b + c \quad (5)$$
$$\forall n.\sum_{i=1}^{n} i^1 = \frac{n \times \mathrm{s}(n)}{2} \quad (6)$$
$$\forall m.2 \times \frac{m}{2} = m \quad (7)$$
$$\forall q.\mathrm{s}(q)^3 = \mathrm{s}(q)^2 + (q \times \mathrm{s}(q)) \times \mathrm{s}(q) \quad (8)$$
$$0 = (0)^2 \quad (9)$$

Fig. 4. Axioms and lemmas assumed in the proof of $\forall n.\sum_{i=1}^{n} i^3 = (\sum_{i=1}^{n} i^1)^2$.

the γ-quantifier for X. This results in the proof state shown on the right hand side of Fig. 3. Thereby $\rho := [U/X, V/Y, Q/P]$, and $\sigma' := [\rho(\sigma(P))/\rho(P)] = [\lambda x.\mathrm{s}(x+0) = \mathrm{s}(x)/Q]$.

Definition 14 (CoRe Calculus). *Let π, π' be* CoRe *proof states. $\pi \vdash_C \pi'$ is a* CoRe proof step *if, and only if, there is a* CoRe *calculus rule $\frac{\pi}{\pi'}$ (cf. Fig. 2). A* CoRe *derivation is a sequence $\pi_1 \vdash_C \pi_2 \vdash_C \ldots \vdash \pi_n$ of* CoRe *proof steps (as usual we define $\vdash_C^*$ as the reflexive transitive closure of $\vdash_C$).*

Let F be a closed formula, $\triangle_I$ the initial extensional expansion tree for F^+, F_X the corresponding QF-formula and id the empty substitution. Then F is provable *in* CoRe*, if, and only if, there is a derivation $\triangle_I; \mathsf{id} \triangleright F_X \vdash_C^* \triangle; \sigma \triangleright \top$.*

Theorem 2 (Soundness & Completeness). *The* CoRe *calculus is sound and complete.*

Proof (Sketch). (Soundness) In order to prove the soundness of the CoRe calculus, we prove (1) for any closed formula F, if F is not valid, then the initial proof state $\triangle_I; \mathsf{id} \triangleright F_X$ for F is satisfiable (by Theorem 1 and Lemma 2); (2) each CoRe calculus rule (except the *Axiom*-rule) preserves the satisfiability of the proof state; (3) proof states on which the CoRe *Axiom*-rule is applicable are unsatisfiable (a single path with $\top^+$ is unsatisfiable by Corollary 1).
(Completeness) By completeness of extensional expansion proofs [15] we can assume for any valid closed formula F that we have guessed the right multiplicities for γ-type quantifiers, the right substitution σ, and the necessary applications of *Leibniz*, *f-Ext*, *b-Ext*, ζ*-Elim*, and *Cut*.[6] All paths in the resulting QF-formula F_P are (propositionally) unsatisfiable. That is from $\triangle_I; Id \triangleright F_X$ we can derive a proof state $\triangle_P; \sigma \triangleright F_P$. In a second phase we show that from $\triangle_P; \sigma \triangleright F_P$ we can derive $\triangle; \sigma \triangleright \top$ by proving that *path resolution* [14] is admissible using *Res*, *Contract*, *Weakening* and the simplification rules. The completeness then finally follows from the completeness of path resolution for propositional logic. □

4 Example

We illustrate the CoRe calculus by proving the following theorem over the natural numbers: $\forall n.\sum_{i=1}^{n} i^3 = (\sum_{i=1}^{n} i)^2$. The function symbols have the standard

[6] *Cut* is used to simulate the extensionality rule from [15]. *Cut* is not admissible in the CoRe-calculus, but would probably be if we add the rules ξ and b (see [5]).

meaning. Besides the necessary definitions and the structural induction axiom for the natural numbers, we assume specific lemmas, in order to keep the proof short. The axioms and lemmas, i.e. the assertions we assume, are given in Fig. 4.

The initial proof state consists of an extensional expansion tree $\triangle_0$ for the conjunction of the assertions implying the conjecture, an empty substitution, and the quantifier free formula resulting from $\triangle_0$. The initial proof state then is

$$\triangle_0; \mathsf{id} \triangleright \left(\begin{array}{ll} (P(0) \wedge (P(\mathrm{y}) \Rightarrow P(\mathrm{s}(\mathrm{y})))) \Rightarrow P(X) & (1)\ A' = B' \Rightarrow A' + C' = B' + C' \quad (5) \\ \sum_{i=1}^{0} i^N = 0 & (2)\ \sum_{i=1}^{N'} i^1 = \frac{N' \times \mathrm{s}(N')}{2} \quad (6) \\ \sum_{i=1}^{\mathrm{s}(M)} i^N = \mathrm{s}(M)^N + \sum_{i=1}^{M} i^N & (3)\ 2 \times \frac{M'}{2} = M' \quad (7) \\ (A+B)^2 = (A^2 + (2 \times B) \times A) + B^2 & (4)\ \mathrm{s}(Q)^3 = \mathrm{s}(Q)^2 + (Q \times \mathrm{s}(Q)) \times \mathrm{s}(Q) \quad (8) \\ & 0 = (0)^2 \quad (9) \end{array} \right) \Rightarrow \sum_{i=1}^{\mathrm{n}} i^3 = (\sum_{i=1}^{\mathrm{n}} i^1)^2$$

Due to lack of space, we introduce a macro step *RuleApplication from* H for the application of some assertion H. Such a macro step consists of applying a replacement rule that can be obtained from H by (1) increasing the multiplicities with respect to the variables that need to be instantiated to apply the rule, (2) apply the necessary substitution, (3) apply the replacement rule, and finally (4) "remove" the instantiated assertion by weakening. The proof is then as follows:

1. By *RuleApplication from* (1)
   $\triangle_1; \sigma_1 \triangleright \left((1) \wedge \sum_{i=1}^{0} i^N = 0 \wedge (3) \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge (8) \wedge (9) \right)$
   $\Rightarrow \sum_{i=1}^{0} i^3 = (\sum_{i=1}^{0} i^1)^2 \wedge (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow (\sum_{i=1}^{\mathrm{s}(y')} i^3 = (\sum_{i=1}^{\mathrm{s}(y')} i^1)^2)$
2. By *RuleApplication from* (2)(2×)
   $\triangle_2; \sigma_2 \triangleright \left((1) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge (8) \wedge 0 = (0)^2 \right)$
   $\Rightarrow 0 = (0)^2 \wedge (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow (\sum_{i=1}^{\mathrm{s}(y')} i^3 = (\sum_{i=1}^{\mathrm{s}(y')} i^1)^2)$
3. By *RuleApplication from* (9)
   $\triangle_3; \sigma_3 \triangleright \left((1) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge (8) \wedge (9) \right)$
   $\Rightarrow \top \wedge (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow (\sum_{i=1}^{\mathrm{s}(y')} i^3 = (\sum_{i=1}^{\mathrm{s}(y')} i^1)^2)$
4. By *Simplify by* $\wedge_L^{\top}$
   $\triangle_4; \sigma_4 \triangleright \left((1) \wedge (2) \wedge \sum_{i=1}^{\mathrm{s}(M)} i^N = \mathrm{s}(M)^N + \sum_{i=1}^{M} i^N \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge (8) \wedge (9) \right)$
   $\Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow (\sum_{i=1}^{\mathrm{s}(y')} i^3 = (\sum_{i=1}^{\mathrm{s}(y')} i^1)^2)$
5. By *RuleApplication from* (3)
   $\triangle_5; \sigma_5 \triangleright \left((1) \wedge (2) \wedge \sum_{i=1}^{\mathrm{s}(M)} i^N = \mathrm{s}(M)^N + \sum_{i=1}^{M} i^N \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge (8) \wedge (9) \right)$
   $\Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow (\mathrm{s}(y')^3 + \sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{\mathrm{s}(y')} i^1)^2)$
6. By *RuleApplication from* (3)
   $\triangle_6; \sigma_6 \triangleright \left((1) \wedge (2) \wedge (3) \wedge (A+B)^2 = (A^2 + (2 \times B) \times A) + B^2 \wedge (5) \wedge (6) \wedge (7) \wedge (8) \wedge (9) \right)$
   $\Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow (\mathrm{s}(y')^3 + \sum_{i=1}^{y'} i^3 = (\mathrm{s}(y') + \sum_{i=1}^{y'} i^1)^2)$
7. By *RuleApplication from* (4)
   $\left((1) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge (8) \wedge (9) \right)$
   $\triangle_7; \sigma_7 \triangleright \Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow \mathrm{s}(y')^3 + \sum_{i=1}^{y'} i^3 =$
   $(\mathrm{s}(y')^2 + (2 \times \sum_{i=1}^{y'} i^1) \times \mathrm{s}(y')) + (\sum_{i=1}^{y'} i^1)^2$
8. By *RuleApplication from* $\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2$
   $\left((1) \wedge (2) \wedge (3) \wedge (4) \wedge A' = B' \Rightarrow A' + C' = B' + C' \wedge (6) \wedge (7) \wedge (8) \wedge (9) \right)$
   $\triangle_8; \sigma_8 \triangleright \Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow \mathrm{s}(y')^3 + (\sum_{i=1}^{y'} i^1)^2 =$
   $(\mathrm{s}(y')^2 + (2 \times \sum_{i=1}^{y'} i^1) \times \mathrm{s}(y')) + (\sum_{i=1}^{y'} i^1)^2$
9. By *RuleApplication from* (5)
   $\triangle_9; \sigma_9 \triangleright \left((1) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge \sum_{i=1}^{N'} i^1 = \frac{N' \times \mathrm{s}(N')}{2} \wedge (7) \wedge (8) \wedge (9) \right)$
   $\Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow \mathrm{s}(y')^3 = \mathrm{s}(y')^2 + (2 \times \sum_{i=1}^{y'} i^1) \times \mathrm{s}(y')$

10. By *RuleApplication from* (6)
    $\triangle_{10}; \sigma_{10} \triangleright \Big((1) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge (6) \wedge 2 \times \frac{M'}{2} = M' \wedge (8) \wedge (9) \Big)$
    $\Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow \mathrm{s}(\mathrm{y}')^3 = \mathrm{s}(\mathrm{y}')^2 + (2 \times \frac{\mathrm{y}' \times \mathrm{s}(\mathrm{y}')}{2}) \times \mathrm{s}(\mathrm{y}')$
11. By *RuleApplication from* (7)
    $\triangle_{11}; \sigma_{11} \triangleright \Big((1) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge \mathrm{s}(Q)^3 = \mathrm{s}(Q)^2 + (Q \times \mathrm{s}(Q)) \times \mathrm{s}(Q) \wedge (9) \Big)$
    $\Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow \mathrm{s}(\mathrm{y}')^3 = \mathrm{s}(\mathrm{y}')^2 + (\mathrm{y}' \times \mathrm{s}(\mathrm{y}')) \times \mathrm{s}(\mathrm{y}')$
12. By *RuleApplication from* (8)
    $\triangle_{12}; \sigma_{12} \triangleright \Big((1) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge (6) \wedge (7) \wedge (8) \wedge (9) \Big) \Rightarrow (\sum_{i=1}^{y'} i^3 = (\sum_{i=1}^{y'} i^1)^2) \Rightarrow \top$
13. By *Simplify by* $2\times \Rightarrow_R^{\top}$: $\triangle_{13}; \sigma_{13} \triangleright \top$
14. By *Axiom*: *q.e.d*

Discussion. The proof illustrates how the CoRe calculus supports direct reasoning at the assertion level. The main focus here is less on the fact that we can represent a proof at the assertion level, but rather on the support offered by the calculus to perform a proof at the assertion level. Indeed, representing such a proof is also possible, for instance, in a natural deduction calculus, provided that it comes with a strong equality substitution rule and that the syntactical structure of the assertions fits the decomposition structure of the calculus rules. However, formulas must typically be decomposed in order to apply the contained knowledge, as for instance in the application of the induction hypothesis in step 8 in the example proof. Moreover, there is no support which actively suggests *how* the assertions can be applied. The CoRe calculus provides this information as replacement rules, which are directly read out from the assertions.

5 Related Work and Conclusion

We presented the CoRe calculus for contextual reasoning. It uses proof theoretic information to statically determine the available assertions for arbitrary subformulas. The proof theoretic information is further used to operationalize the application of the assertions via replacement rules. Therefore the CoRe calculus actively supports direct reasoning on the assertion level.

The technical details of the calculus, for instance, that replacement rules are in principle non-normalform resolution and paramodulation rules, can mostly be hidden from the user. However, they should ease the implementation of automated proof procedures, and any proof constructed on their basis would immediately be available as an assertion level proof, which should ease the understanding of the proofs for a human user. Although in this paper we focused on higher-order logic, in [3] we define CoRe calculi on top of the matrix calculi for most of the first-order modal logics considered in [18].

The CoRe calculus should also provide a suitable basis to accommodate a deduction modulo [8] approach, since it allows to implement a *contextual* congruence on propositions which takes the logical context of subformulas into account through the uniform mechanism to synthesize all available rules from it. It also is in the Deep Inference paradigm for calculi, which is studied for instance in the Calculus of Structures [6]. However, rather than studying proof theoretic properties using deep structural manipulation rules, the CoRe calculus aims at

supporting a reasoning based on assertions through replacement rules. Finally, it is also related to Focusing Proof Construction [1] which derives sequent calculus macro steps from available assertions. Unlike replacement rules, the focusing proof steps can only be determined for top-level formulas, only be applied to top-level formulas, and do not include conditional equivalences and equations.

Future work consists of treating equality as a primitive concept, define CoRe calculi for further logics (especially intuitionistic logics), provide transformations of CoRe proofs into sequent calculus proofs to ease proof checking, and investigate automated reasoning procedures for CoRe by extending ordering-based automated reasoning techniques like [10] to non-normalform resolution and paramodulation.

References

1. J.-M. Andreoli. Focusing and proof construction. *Annals of Pure and Applied Logic*, 107(1):131–163, 2000.
2. P. B. Andrews. General models, descriptions, and choice in type theory. *The Journal of Symbolic Logic*, 37(2):385–397, June 1972.
3. S. Autexier. *Hierarchical Contextual Reasoning*. PhD thesis, Computer Science Department, Saarland University, Saarbrücken, Germany, 2003.
4. H.Barendregt. *The λ-Calculus - Its Syntax and Semantics*. North Holland, 1984.
5. C. Benzmüller, C. E. Brown, and M. Kohlhase, Semantic Techniques for Higher-Order Cut-Elimination. SEKI Tech. Report SR-2004-07, Saarland University, 2004.
6. K. Brünnler. *Deep Inference and Symmetry in Classical Proofs*. Logos, 2004.
7. A. Bundy. The use of explicit plans to guide inductive proofs. DAI Research Report 349, Department of Artificial Intelligence, University of Edinburgh, 1987.
8. G. Dowek, Th. Hardin, and C. Kirchner. Theorem proving modulo. Rapport de Recherche 3400, INRIA, April 1998.
9. M. Fitting. Tableau methods of proof for modal logics. *Notre Dame Journal of Formal Logic*, XIII:237–247, 1972.
10. H. Ganzinger and J. Stuber. Superposition with equivalence reasoning and delayed clause normal form transformation. In F. Baader, editor, *Automated Deduction — CADE-19*, volume 2741 of *LNCS*, pages 335–349. Springer-Verlag, 2003.
11. M. J. Gordon, A. J. Milner, and C. P. Wadsworth. *Edinburgh LCF – A mechanised logic of computation*. Springer Verlag, 1979. LNCS 78.
12. L. Henkin. Completeness in the theory of types. *The Journal of Symbolic Logic*, 15:81–91, 1950.
13. X. Huang. *Human Oriented Proof Presentation: A Reconstructive Approach*. Number 112 in DISKI. Infix, Sankt Augustin, Germany, 1996.
14. N. V. Murray and E. Rosenthal. Inference with path resolution and semantic graphs. *Journal of the Association of Computing Machinery*, 34(2):225–254, 1987.
15. F. Pfenning. *Proof Transformation in Higher-Order Logic*. Phd thesis, Carnegie Mellon University, 1987.
16. K. Schütte. *Proof Theory. (Originaltitel: Beweistheorie)*, volume 255 of *Die Grundlehren der mathematischen Wissenschaften*. Springer, 1977.
17. R. M. Smullyan. *First-Order Logic*, volume 43 of *Ergebnisse der Mathematik*. Springer-Verlag, Berlin, 1968.
18. L. Wallen. *Automated proof search in non-classical logics: efficient matrix proof methods for modal and intuitionistic logics*. MIT Press series in AI, 1990.