Extensional Higher-Order Resolution

Christoph Benzmüller and Michael Kohlhase

Fachbereich Informatik, Universität des Saarlandes, Germany
chris|kohlhase@cs.uni-sb.de

Abstract. In this paper we present an extensional higher-order resolution calculus that is complete relative to Henkin model semantics. The treatment of the extensionality principles – necessary for the completeness result – by specialized (goal-directed) inference rules is of practical applicability, as an implentation of the calculus in the Leo-System shows. Furthermore, we prove the long-standing conjecture, that it is sufficient to restrict the order of primitive substitutions to the order of input formulae.

1 Introduction

The history of building automated theorem provers for higher-order logic is almost as old as the field of deduction systems itself. The first successful attempts to mechanize and implement higher-order logic were those of Huet [Hue73] and Jensen and Pietrzykowski [JP76]. They combine the resolution principle for higher-order logic (first studied in [And71]) with higher-order unification. The unification problem in typed λ-calculi is much more complex than that for first-order terms, since it has to take the theory of $\alpha\beta\eta$-equality into account. In particular the higher-order unification problem is undecidable and sets of solutions need not to have most general elements that represent them. Thus the calculi for higher-order logic have to take special measures to circumvent the problems posed by the theoretical complexity of higher-order unification.

Experiments like the Tps system [And89,ABI$^+$96] (which uses a higher-order matings calculus) or our own Leo system [BK98,Ben97] (which uses a variant of Huet's resolution calculus [Hue73]) have shown the practical feasibility of higher-order automated theorem proving based on these ideas. Establishing completeness for higher-order calculi is more problematic than in first-order logic. The intuitive set-theoretic *standard semantics* cannot give a sensible notion of completeness, since it does not admit complete calculi [Göd31]. But there is a more general notion of semantics due to Henkin [Hen50] that allows complete calculi and therefore sets the standard for the deductive power of calculi.

The core of higher-order resolution ($\mathcal{HORES}$, see [Hue73,Koh94] for details) is a simple extension of the first-order resolution method to the higher-order language: the only significant difference is that $\beta\eta$-equality has to be build in by keeping formulae in normal form and that first-order unification has to be replaced by higher-order unification (i.e. unification with respect to the theory of $\beta\eta$-equality). Since this is a semi-decidable search process itself, it cannot simply be used as a sub-procedure that is invoked during the application of the resolution or factoring rules. Rather resolution and factorization rules are modified, so that they record the induced unification problem in a unification constraint instead of trying to compute a complete set of unifiers. Furthermore, the calculus is augmented with the inference rules of higher-order unification that are lifted to act on the unification constraints of clauses. With this trick the search for empty clauses and that for higher-order unifiers are interleaved, which alleviates the undecidability problem.

Unfortunately, neither $\mathcal{HORES}$ nor the TPS procedure are complete with respect to Henkin semantics, since they fail to capture substitutivity of equivalence. In [Koh95], the first author has presented a higher-order tableau calculus that addresses the problem with a new inference rule that uses substitutivity of equivalence in a goal-oriented way, but still fails to capture functional extensionality of Leibniz equality.

For our extensional higher-order resolution calculus $\mathcal{ER}$ we extend higher-order resolution by ideas from [Koh95] and a suitable treatment of Leibniz equality and prove the resulting calculus sound and complete with respect to Henkin's general model semantics [Hen50]. Furthermore, we show that we can restrict the set of primitive substitutions that are necessary for flexible literals to a finite set.

Before we begin with the exposition, let us specify what we mean by "higher-order logic": any simply typed logical system that allows quantification over function variables. In this paper, we will employ a system $\mathcal{HOL}$, which is based on the simply typed λ-calculus; for an introduction see for instance [And86,Bar84].

2 Higher-Order Logic ($\mathcal{HOL}$)

The set $wff_\alpha(\Sigma)$ of well-formed formulae of type α is build up from the set $\mathcal{V}$ of variables, and the signature Σ (a set of typed constants) as applications and λ-abstractions. We will denote variables with upper-case letters $(X_\alpha, Y, Z, X_\beta^1, X_\gamma^2 \ldots)$, constants with lower-case letters $(c_\alpha, f_{\alpha\to\beta}, \ldots)$, and well-formed formulae with upper-case bold letters $(\mathbf{A}_\alpha, \mathbf{B}, \mathbf{C}^i, \ldots)$[1]. Furthermore, we abbreviate multiple applications and abstractions in a kind of vector notation, so that $\mathbf{A}\overline{\mathbf{U}^k}$ denotes k-fold application (associating to the left) and $\lambda\overline{X^k}.\mathbf{A}$ denotes k-fold λ-abstraction (associating to the right) and use the square dot **.** as an abbreviation for a pair of brackets, where **.** stands for the left one with its partner as far to the right as is consistent with the bracketing already present in the formula.

We will use the terms like free and bound variables in their standard meaning and we use $\mathbf{Free}(\mathbf{A})$ for the set of free variables of a formula $\mathbf{A}$. In particular alphabetic change of names of bound variables is build into our $\mathcal{HOL}$: we consider alphabetic variants to be identical (viewing the actual representation as a representative of an alphabetic equivalence class) and use a notion of substitution that avoids variable capture, systematically renaming bound variables. We could also have used de Bruijn's indices [dB72] as a concrete implementation of this approach at the syntax level.

[1] We will denote the types of formulae as indices, if it is not clear from the context.

By $\mathit{wff}^{cl}_\alpha(\Sigma) \subseteq \mathit{wff}_\alpha(\Sigma)$ we denote the set of all closed well-formed formulae, i.e. which contain no free variables and we call the members of $\mathit{wff}_o(\Sigma)$ sentences.

We denote a substitution that instantiates a variable X with a formula $\mathbf{A}$ with $[\mathbf{A}/X]$ and write $\sigma, [\mathbf{A}/X]$ for the substitution that is identical with σ but instantiates X with $\mathbf{A}$.

The structural equality relation of $\mathcal{HOL}$ is induced by $\beta\eta$-reduction

$$(\lambda X\mathbf{.A})\mathbf{B} \longrightarrow_\beta [\mathbf{B}/X]\mathbf{A} \qquad\qquad (\lambda X\mathbf{.C}X) \longrightarrow_\eta \mathbf{C}$$

where X is not free in $\mathbf{C}$. It is well-known, that the reduction relations β, η, and $\beta\eta$ are terminating and confluent, so that there are unique normal forms.

In $\mathcal{HOL}$, the set of base types is $\{o, \iota\}$ for truth values and individuals, and the signature Σ contains logical constants for negation $\neg_{o\to o}$, conjunction $\wedge_{o\to o\to o}$, and quantification[2] $\Pi^\alpha_{(\alpha\to o)\to o}$. All other constants are called parameters, since the argumentation in this paper is parametric in their choice[3].

It is matter of folklore that equality can directly be expressed in $\mathcal{HOL}$ e.g. by the *Leibniz definition*, so that a primitive notion of equality (expressed by a primitive constant $=$ in Σ) is not strictly needed; we will use this observation in this paper to treat equality as a defined notion. Leibniz equality defines two terms to be equal, iff they have the same properties. Hence equality can be defined as

$$\doteq^\alpha \;:= \lambda X_\alpha\mathbf{.}\lambda Y_\alpha\mathbf{.}\forall P_{\alpha\to o}\mathbf{.}PX \Rightarrow PY$$

A **standard model** for $\mathcal{HOL}$ provides a fixed set $\mathcal{D}_\iota$ of individuals, and a set $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ of truth values. All the domains for the complex types are defined inductively: $\mathcal{D}_{\alpha\to\beta}$ is the set of functions $f\colon \mathcal{D}_\alpha \to \mathcal{D}_\beta$. The evaluation $\mathcal{I}_\varphi$ with respect to an interpretation $\mathcal{I}\colon \Sigma \to \mathcal{D}$ of constants and an assignment φ of variables is obtained by the standard homomorphic construction that evaluates a λ-abstraction with a function, whose operational semantics is specified by β-reduction.

Henkin models only require that $\mathcal{D}_{\alpha\to\beta}$ has enough members that any well-formed formula can be evaluated[4]. Note that with this generalized notion of a model, there are less formulae that are valid in all models (intuitively, for any given formulae there are more possibilities for counter-models). Thus the generalization to Henkin models restricts the set of valid formulae sufficiently, so that all of them can be proven by the resolution calculus presented in this paper. For our completeness proofs, we will use the abstract consistency method first introduced by Raymond Smullyan in [Smu63] for first-order logic and later extended to higher-order logic by Peter Andrews [And71]. The model existence theorem below is a variant of the latter for Henkin models. For the proof we refer to [BK97].

[2] With this quantification constant, standard quantification of the form $\forall X_\alpha\mathbf{.A}$ can be regained as an abbreviation for $\Pi^\alpha(\lambda X_\alpha\mathbf{.A})$.

[3] In particular, we do not assume the existence of description or choice operators. For a detailed discussion of the semantic issues raised by the presence of these logical constants see [And72].

[4] In other words: the functional universes are rich enough to satisfy the comprehension axioms.

Theorem 1 (Henkin Model Existence). *Let Γ_Σ be a saturated abstract consistency class for Henkin models (see the definition below), and $\Phi \in \Gamma_\Sigma$, then there is a Henkin model $\mathcal{M}$ such that $\mathcal{M} \models \Phi$.*

Definition 1 (Abstract Consistency Class for Henkin Models). *We call a class Γ_Σ of sets of sentences an* **abstract consistency class for Henkin Models**, *iff Γ_Σ is closed under subsets and such that for all sets $\Phi \in \Gamma_\Sigma$ (we use $\Phi * \mathbf{A}$ as an appreviation for $\Phi \cup \{\mathbf{A}\}$):*

∇_c *If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.*
$\nabla_\neg$ *If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.*
$\nabla_{\beta\eta}$ *If $\mathbf{A} \in \Phi$ and $\mathbf{B}$ is the $\beta\eta$-normal form of $\mathbf{A}$, then $\mathbf{B} * \Phi \in \Gamma_\Sigma$.*
$\nabla_\vee$ *If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \mathbf{B} \in \Gamma_\Sigma$.*
$\nabla_\wedge$ *If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi * \neg\mathbf{A} * \neg\mathbf{B} \in \Gamma_\Sigma$.*
$\nabla_\forall$ *If $\Pi^\alpha\mathbf{F} \in \Phi$, then $\Phi * \mathbf{F}\mathbf{G} \in \Gamma_\Sigma$ for each $\mathbf{G} \in wff^{cl}_\alpha(\Sigma)$.*
$\nabla_\exists$ *If $\neg\Pi^\alpha\mathbf{F} \in \Phi$, then $\Phi * \neg(\mathbf{F}w) \in \Gamma_\Sigma$ for a fresh parameter $w_\alpha \in \Omega_\alpha$.*
∇_b *If $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$, then $\Phi \cup \{\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$ or $\Phi \cup \{\neg\mathbf{A}, \mathbf{B}\} \in \Gamma_\Sigma$.*
∇_q *If $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}) \in \Phi$, then $\Phi * \neg(\mathbf{F}w \doteq^\beta \mathbf{G}w) \in \Gamma_\Sigma$ for a fresh parameter $w_\alpha \in \Omega_\alpha$.*

We will call Γ_Σ **saturated**, *iff for all sentences $\mathbf{A} \in wff_o(\Sigma)$ we have $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} \in \Gamma_\Sigma$.*

Remark 1 (Counterparts for ∇_b,∇_q). In Definition 1 positive counterparts for the two conditions ∇_b,∇_q are not needed, since these conditions are automatically met (note that $\doteq$ is a defined construct). For details see [BK97].

In this paper the *extensionality principles* will play a major role. These formalize fundamental mathematical intuitions about functions and truth values. The **functional extensionality principle** says, that two functions are equal, iff they are equal on all arguments. This principle can be formulated by the following schematic λ-term:

$$\forall M_{\alpha\to\beta}.\forall N_{\alpha\to\beta}.(\forall X.(MX) \doteq (NX)) \equiv (M \doteq N)$$

The **extensionality principle for truth values** states that on the set of truth values equality and equivalence relation coincide: $\forall P_o.\forall Q_o.(P \doteq Q) \equiv (P \equiv Q)$. Note that in Henkin models both extensionality principles are valid and that Leibniz equality indeed denotes equality relation (see [BK97] for details).

3 The Calculus $\mathcal{ER}$

Now we introduce the higher-order resolution calculus $\mathcal{ER}$. Therefore we will review standard higher-order resolution $\mathcal{HORES}$ and use the extensionality principles to discuss why it is not complete. From the deficiencies we will develop the necessary extensions and give an intuition by exhibiting refutations that become possible.

$\mathcal{HORES}$ is a refutation calculus that manipulates sets of *clauses*, i.e. sets (which we will represent as disjunctions) of literals (e.g. $\mathcal{C} := [q_{\alpha\to o}X_\alpha]^T \vee [p_{\alpha\to o}X_\alpha]^F \vee [c_\alpha = X_\alpha]^F$).

Definition 2 (Literal). *Literals are atomic propositions labeled with an intended truth value. We call a literal a* unification constraint*, iff it is negative (i.e. annotated by the truth value F) and the head is $=$, all the others we call* **proper literals***. Clauses existing entirely of unification constraints are called* almost empty*. Since instantiation of a head variable will convert a literal into a general labeled propositions, we will sometimes call these* **pre-literals***.*

Clause normalization is very similar to the first-order case, except for the treatment of existential quantification. Therefore, we will not present the transformation rules here, but simply discuss the differences and assume that each given higher-order proof problem $\mathcal{P}$ can be transformed into a set of clauses $\mathrm{CNF}(\mathcal{P})$. A naive treatment with Skolemization results in a calculus that is not sound with respect to Henkin models, since Skolem functions are special choice functions[5], which are not guaranteed to exist in Henkin models. A solution due to [Mil83] is to associate with each Skolem constant the minimum number of arguments the constant has to be applied to. Skolemization becomes sound, if any Skolem function f^n only occurs in a *Skolem term*, i.e. a formula $\mathbf{S} = f^n\overline{\mathbf{A}^n}$, where none of the $\mathbf{A}^i$ contains a bound variable. Thus the Skolem terms only serve as descriptions of the existential witnesses and never appear as functions proper. When we speak of a **Skolem term $\mathbf{S}_\alpha$ for a clause** C, where $\{X^1_{\alpha^1} \cdots X^n_{\alpha^n}\}$ is the set of free variables occurring in C, then $\mathbf{S}_\alpha$ is an abbreviation for the term $(f^n_{\alpha^1 \to \cdots \to \alpha^n \to \alpha} X^1 \cdots X^n)$, where f is a new constant from $\mathcal{C}_{\alpha^1 \to \cdots \to \alpha^n \to \alpha}$ and n specifies the number of necessary arguments for f.

Remark 2 (Leibniz Equality). We assume that before applying clause normalization each primitive equality symbol is replaced by its corresponding Leibniz definition. Hence after normalizing a given input problem, the resulting clause set does not contain any equality symbol. However, during the refutation process, equality symbols may be introduced again as we code unification constraints by negated equation literals.

3.1 Higher-Order Unification in $\mathcal{ER}$

Higher-order unification is a process of recursive deterministic simplification (rules α, η, *Dec*, *Triv*, and *Subst* in figure 1) and non-deterministic variable binding (rule *Flex/Rigid*). The rules α and η are licensed by the functional extensionality principle and eliminate the top λ-binder in unification constraints of functional type. The Skolem term s_α is an existential witness for the fact that the functions are different. Since clauses are implicitly universally quantified, this witness may depend on the values of all free variables occurring in the clauses, so it must be a Skolem term for this clause. Decomposition (rule *Dec*) is analogous to the first-order case and the rule *Triv* allows to remove reflexivity pairs. Rule *Dec* will be discussed again in connection with the extensionality rules in section 3.3.

The rule *Subst* eliminates variables that are solved in a clause: we call a unification constraint $U := [X_\alpha = \mathbf{N}_\alpha]^F$ or $U := [\mathbf{N}_\alpha = X_\alpha]^F$ **solved** iff X_α is

[5] They choose an existential witness from the set of possible witnesses for an existential formula.

$$\frac{C \lor [(\lambda X_\alpha.\mathbf{A}) = (\lambda Y_\alpha.\mathbf{B})]^F \qquad s_\alpha \text{ Skolem term for this clause}}{C \lor [[s/X]\mathbf{A} = [s/Y]\mathbf{B}]^F}\ \alpha$$

$$\frac{C \lor [(\lambda X_\alpha.\mathbf{A}) = \mathbf{B}]^F \qquad s_\alpha \text{ Skolem term for this clause}}{C \lor [[s/X]\mathbf{A} = (\mathbf{B}s)]^F}\ \eta$$

$$\frac{C \lor [h\overline{\mathbf{U}^n} = h\overline{\mathbf{V}^n}]^F}{C \lor [\mathbf{U}^1 = \mathbf{V}^1]^F \lor \ldots \lor [\mathbf{U}^n = \mathbf{V}^n]^F}\ \mathit{Dec} \qquad \frac{C \lor [\mathbf{A} = \mathbf{A}]^F}{C}\ \mathit{Triv}$$

$$\frac{C \lor E \quad E \text{ solved for } C}{\mathrm{CNF}(\mathrm{subst}_E(C))}\ \mathit{Subst}$$

$$\frac{C \lor [F_\gamma\overline{\mathbf{U}^n} = h\overline{\mathbf{V}}]^F \quad \mathbf{G} \in \mathcal{GB}^h_\gamma}{C \lor [F = \mathbf{G}]^F \lor [F\overline{\mathbf{U}} = h\overline{\mathbf{V}}]^F}\ \mathit{Flex/Rigid}$$

Fig. 1. Lifted Higher-Order (pre-)unification rules

not free in $\mathbf{N}_\alpha$. In this case X is called the **solved variable** of U. Let $C := L^1 \lor \cdots \lor L^n \lor U^1 \lor \cdots \lor U^m$ be a clause with unification constraints $U^1 \lor \cdots \lor U^m$ $(1 \leq m)$. Then a disjunction $U^{i_1} \lor \cdots \lor U^{i_k}$ $(i_j \in \{1, \cdots, m\}; 1 \leq j \leq k)$ of solved unification constraints occurring in C is called **solved for** C iff for every $U^{i_j}(1 \leq j \leq k)$ holds: the solved variable of U^{i_j} does not occur free in any of the U^{i_l} for $l \neq j; 1 \leq l \leq k$. Note that each solved set of unification constraints E for a clause C can be associated with a substitution subst_E which is the most general unifier of E. Thus the rule *Subst* essentially propagates the information from the unification constraints to the proper clause parts. Since the instantiation of flexible literals (i.e. literals, where the head is a free variable) may result in pre-literals, the result of this propagation may cease to be a clause, therefore it needs to be reduced to clause normal form.

Remark 3 (Eager Unification). The set of rules described up to now is terminating and confluent, so that higher-order unification applies it eagerly to filter out all clauses with an unsolvable unification constraint[6]. It leads to unification constraints, where both sides are applications and where at least one side is flexible, i.e. where the head is a variable. In this case, the higher-order unification problem can be reduced to the problem of finding most general formulae of a given type and a given head symbol.

Definition 3 (General Binding). *Let $\alpha = (\overline{\beta^l} \to \gamma)$, and h be a constant or variable of type $(\overline{\delta_m} \to \gamma)$ in Γ, then $\mathbf{G} := \lambda\overline{X^l_{\beta^l}}.h\overline{\mathbf{V}^m}$ is called a* **general**

[6] As we will see later this solution is too strong if we want to be complete in Henkin models since an unsolvable unification constraint might be solvable by using the extensionality rules.

binding of type α **and head** h, *if* $\mathbf{V}^i = H^i\overline{X^l_{\beta_l}}$. *The* H^i *are new variables of types* $\overline{\beta^l} \to \delta^i]$. *It is easy to show that general bindings indeed have the type and head claimed in the name and are most general in the class of all such terms.*

General bindings, where the head is a bound variable $X^j_{\beta_j}$ *are called* **projection bindings** *(we write them as* $\mathcal{G}^j_\alpha$*) and* **imitation bindings** *(written* $\mathcal{G}^h_\alpha$*) else. Since we need both imitation and projection bindings for higher-order unification, we collect them in the set of* **approximating bindings for** h **and** α *(*$\mathcal{GB}^h_\alpha := \{\mathcal{G}^h_\alpha\} \cup \{\mathcal{G}^j_\alpha \mid j \leq l\}$*).*

Since there are only finitely many general bindings (one imitation binding and at most l projection bindings) the *Flex/Rigid* rule is finitely branching. We never have to consider the so-called *Flex/Flex* literals[7], since *Flex/Flex* equations can always be solved by instantiating the head variables with suitable constant functions that absorb their arguments. This observation is due to Gérard Huet [Hue73] and defines higher-order pre-unification, a computationally more feasible (but still undecidable) variant of higher-order unification. However, even if *Flex/Flex* pairs are solvable, we cannot simply delete them like trivial pairs, since one or both of the heads may be instantiated making the term rigid, so that the pair has to be subject to pre-unification again.

3.2 Higher-Order resolution

Definition 4 (Higher-Order Resolution). *The* **higher-order resolution calculus** $\mathcal{HORES}$ *consists of the inference rules in figure 2 together with the unification rules in figure 1. We call a clause* **empty**, *iff it consists entirely of Flex/Flex unification constraints and say hat a* $\mathcal{HORES}$*-derivation of an empty clause from a set* Φ *of clauses is a* **refutation** *of* Φ. *For a sentence* $\mathbf{A}_o$ *we call a refutation of CNF*$(\neg\mathbf{A})$ *a refutation for* $\mathbf{A}$.

As in first-order we have resolution and factorization rules *Res* and *Fac*. But instead of solving the unification problems immediately within a rule application we delay their solution and incorporate them explicitly as unification constraints in the resulting clauses. Note that the resolution rule as well as the factorization rule are allowed to operate on unification constraints.

To find a refutation for a given problem we may have to instantiate the head variables of flexible literals by material that contains logical constants. Unfortunately these instantiations cannot be generated by the unification rules, since all logical constants have been eliminated from the clause set by normalization, thus they enter the refutation by unification. Therefore the rule *Prim* allows to instantiate head variables Q_γ by general bindings $\mathbf{P}$ of type γ and head in $\{\neg, \vee\} \cup \{\Pi^\beta | \beta \in \mathcal{T}\}$. Thus the necessary logical constants are introduced into the refutation one by one, hence the name *primitive substitutions*.

For instance the sentence $\mathbf{A} := \exists X_o.X$ is valid in all Henkin models, but $\mathrm{CNF}(\neg\mathbf{A}) = \{[X]^F\}$ cannot be refuted without some kind of a primitive substitution rule, since none of the other rules apply. With *Prim*, we can deduce

[7] For a refutation, we do not need to enumerate all unifiers for a given unification problem but to seek for one possible instantiation of a given problem which leads to the contradiction.

$$\frac{[\mathbf{N}]^\alpha \lor C \quad [\mathbf{M}]^\beta \lor D \quad \alpha \neq \beta}{C \lor D \lor [\mathbf{N} = \mathbf{M}]^F}\, Res \qquad \frac{[\mathbf{N}]^\alpha \lor [\mathbf{M}]^\alpha \lor C \quad \alpha \in \{T, F\}}{[\mathbf{N}]^\alpha \lor C \lor [\mathbf{N} \ = \mathbf{M}]^F}\, Fac$$

$$\frac{[Q_\gamma \overline{\mathbf{U}^k}]^\alpha \lor C \quad \mathbf{P} \in \mathcal{GB}_\gamma^{\{\neg,\lor\}\cup\{\Pi^\beta \mid \beta \in \mathcal{T}^k\}}}{[Q_\gamma \overline{\mathbf{U}^k}]^\alpha \lor C \lor [Q = \mathbf{P}]^F}\, Prim^k$$

Fig. 2. Higher-order resolution rules

$[X]^F \lor [X = \neg H]^F$ and then $[Y]^T$ by *Subst.* These two unit literals can be resolved to $[X = Y]^F$, which is an empty clauses, since $[X = Y]^F$ is a *Flex/Flex* unification constraint.

The primitive substitution rules have originally been introduced by Peter Andrews in [And89] (Gérard Huet uses a set of so-called "splitting rules" for the same purpose in [Hue73]). Note that the set of general bindings is infinite, since we need one for every quantifier Π^α and the set of types is infinite. Thus in contrast to the goal-directed search for instantiations in unification, the rule *Prim* performs blind search and even worse, is infinitely branching. Therefore, the problem of finding instantiations for predicate variables is conceived as the limiting factor to higher-order automated theorem proving.

It has been a long-standing conjecture that in machine-oriented calculi it is sufficient to restrict the order of primitive quantifier substitutions to the order of the input formulae. In [BK97], we have established a finer-grained variant of theorem 1 that we can use as a basis to prove this conjecture. Let us now introduce the necessary definitions.

Definition 5 (Order). *For a type* $\alpha \in \mathcal{T}$, *we define the* **order** $\mathbf{ord}(\alpha)$ *of* α *as* $\mathbf{ord}(\iota) = \mathbf{ord}(o) = 0$, *and* $\mathbf{ord}(\alpha \to \beta) = \max\{\mathbf{ord}(\alpha), \mathbf{ord}(\beta)\} + 1$. *Note that the set* $\mathcal{T}^k = \{\alpha \in \mathcal{T} \mid \mathbf{ord}(\alpha) \leq k\}$ *is finite for any order* k. *We will take the order of a formula to be the highest order of any type of any of its subterms, and the order of a set of formulae to be the maximum of the orders of its members.*

Theorem 2 (Model Existence with Order). *The model existence theorem holds even if we weaken the condition* $\nabla_\forall$ *of an abstract consistency class to*

$\nabla_\forall^k$ *If* $\Pi^\alpha \mathbf{F} \in \Phi$, *then* $\Phi * \mathbf{FG} \in \Gamma_\Sigma$ *for each* $\mathbf{G} \in wff_\alpha^{cl}(\Sigma)$ *with* $\mathbf{ord}(\mathbf{G}) \leq \mathbf{ord}(\Phi)$.

In [BK97] we establish this theorem for arbitrary well-founded orderings on types such that $\mathbf{ord}(\alpha), \mathbf{ord}(\beta) \leq \mathbf{ord}(\alpha \to \beta)$. This allows us to restrict instantiation in $\mathcal{ER}$ to formulae of the order of the input formulae. Note that this only effects the primitive substitution rule, since all other instantiations are performed by unification, which is order-restricted by construction. In particular, the non-standard definition of order above ensures finite branching of the primitive substitution rule. This ordering, that takes the lengths of argument lists into account leads to an increased order of the input set compared to the standard definition of order ($\mathbf{ord}(\overline{\alpha_n} \to \beta) = \max_n\{\alpha_i\} + 1$) and effectively restricts the number of necessary instantiations.

Our result justifies the practice of higher-order theorem provers to restrict the search for primitive substitutions and gives a road-map towards complete procedures. Of course there is still a lot of room for experimentation with the respective orderings.

3.3 Extensionality

The higher-order resolution calculus $\mathcal{HORES}$ defined above is not complete with respect to Henkin models, as the following example will show.

Example 1. The following formulae E1-E5[8] are not provable in $\mathcal{HORES}$ without using additional axioms for functional extensionality and/or extensionality on truth values.

E1 $a_o \equiv b_o \Rightarrow (\forall P_{o\to o}.Pa \Rightarrow Pb)$
This is the non-trivial direction of the extensionality property for truth values: if a_o is equivalent to b_o then a_o is equal to b_o ($a_o \equiv b_o \Rightarrow a = b$).

E2 $\forall P_{o\to o}.P(a_o \wedge b_o) \Rightarrow P(b \wedge a)$.
Any property which holds for $a \wedge b$ also holds for $b \wedge a$ (or simply that $a \wedge b = b \wedge a$).

E3 $(p_{o\to o}a_o \wedge pb_o) \Rightarrow p(b \wedge a)$
In other words, an arbitrary property $p_{o\to o}$ which coincidently holds for a_o and b_o also holds for their conjunction.

E4 $(\forall X_\iota.\forall P_{\iota\to o}.(P(m_{\iota\to\iota}X) \Rightarrow P(n_{\iota\to\iota}X))) \Rightarrow (\forall Q_{(\iota\to\iota)\to o}.Q(\lambda X_\iota.mX) \Rightarrow Q(\lambda X_\iota.nX))$
This formula can be interpreted as an instance of the ξ-rule $(\forall X_\iota.m_{\iota\to\iota}X = n_{\iota\to\iota}X) \Rightarrow (\lambda X_\iota.mX) = (\lambda X_\iota.nX)$ (See for instance [Bar84]).

E5 $(\forall X_\iota.\forall P_{\iota\to o}.P(m_{\iota\to\iota}X) \Rightarrow P(n_{\iota\to\iota}X)) \Rightarrow (\forall Q_{(\iota\to\iota)\to o}.Qm \Rightarrow Qn)$
This is an instance of the non-trivial direction of the functional extensionality axiom for type $\iota \to \iota$: $(\forall X_\iota.(m_{\iota\to\iota}X) = (n_{\iota\to\iota}X) \Rightarrow m = n)$.

For a proof of **E1** note that the clause normal form of the succedent consists of the two unit clauses $[p^0a]^F$ and $[p^0b]^T$, where p^0 is the Skolem constant for the variable P. These can be resolved upon to obtain the clause $[p^0a = p^0b]^F$, which can be decomposed to $[a_o = b_o]^F$. Obviously, this unification constraint cannot be solved by higher-order unification, and hence the refutation fails. In this situation, we need the principle of extensionality on truth values, which allows to replace each negated equality on type o by an equivalence. This leads to the clause normal form of $[a_o \equiv b_o]^F$, which contradicts the antecedent of **E1** and finally gives us the refutation.

Similar investigations show that the other examples cannot be proven by $\mathcal{HORES}$ too.

Our aim is to find an extension of $\mathcal{HORES}$, which is both Henkin-complete and adequate for an implementation. Surely, the introduction of axioms for the

[8] In Problems E1, E2, E4, and E5 we have used Leibniz definition of equality to remove the intuitive equality symbols.

$$\frac{C \vee [\mathbf{M}_o = \mathbf{N}_o]^F}{\mathrm{CNF}(C \vee [\mathbf{M}_o \equiv \mathbf{N}_o]^F)}\ \mathit{Equiv} \qquad \frac{C \vee [\mathbf{M}_\alpha = \mathbf{N}_\alpha]^F \quad \alpha \in \{o, \iota\}}{\mathrm{CNF}(C \vee [\forall P_{\alpha \to o}\text{.}PM \Rightarrow PN]^F)}\ \mathit{Leib}$$

$$\frac{C \vee [\mathbf{M}_{\alpha \to \beta} = \mathbf{N}_{\alpha \to \beta}]^F \quad s_\alpha \text{ Skolem term for this clause}}{C \vee [\mathbf{M}s = \mathbf{N}s]^F}\ \mathit{Func}$$

Fig. 3. Extensionality rules

extensionality principles can solve the completeness problem in theory, but this will lead to an explosion of the search space which has to be avoided in practice. In particular, we do not change the purely negative spirit of the resolution calculus by introducing axioms but introduce special inference rules.

Definition 6 (Extensional Higher-Order Resolution). *The* **extensional higher-order resolution** *calculus $\mathcal{ER}$ is $\mathcal{HORES}$ extended with the inference rules in figure 3.*

The Rule *Leib* instantiates the equality symbol by its Leibniz definition and applies clause normalization. Rule *Equiv* is directly motivated by the proof attempt of **E1** discussed in example 1. Thus rule *Equiv* reflects the extensionality property for truth values but in a negative way: if two formulas are not equal then they are also not equivalent. Rule *Func* does the same for functional extensionality: if two functions are not equal then there exists an argument s_α on which these functions differ. To ensure soundness s_α has to be a new Skolem term which contains all the free variables occurring in the given clause.

The new rules strongly connect the unification part of our calculus with the resolution part. In some sense, they make the unification part extensional, since they allow to modify unification problems, which are not solvable by pre-unification alone in an extensional appropriate way and to translate them back into usual literals, such that we can try to find the right argumentation for the solvability of the unification constraints in the general refutation process by possibly respecting the additionally given clauses in the search space.

Remark 4 (Rule Func). Note that we have already introduced two rules – α and η in unification (see figure 1) – which are very similar to this one. In fact we can restrict rule *Func* to the case were **N** and **M** are non-abstractions or vice-versa, we can remove the α and η rules from simplification as they are subsumed by the rule *Func* as purely type-based and apply β-reduction to both sides of the modified unification constraint.

Remark 5 (Unification Constraints). We have lifted the unification constraints to clause level by coding them into negated equation literals. Hence the question arises whether or not resolution and factorization rules are allowed to be applied on these unification constraints. In order to obtain a Henkin complete calculus this is not necessary – as our completeness proof shows – if we add the three extensionality rules discussed in the next subsection. Consequently the unification constraints do not necessarily have to be coded as negative equation literals, any other form will work as well.

The coding of unification constraints as negated equation literals becomes important if one considers an alternative version of extensional higher order resolution – which we will also motivate below –, where the rule *Leib* is avoided.

Note that none of the three new extensionality rules introduces any flexible literal and even better, they introduce no new free variable at all; even if they heavily increase the search space for refutations, they behave much better – as experiments show with the LEO theorem prover [BK98,Ben97] – than the extensionality axioms, which introduce lots of flexible literals in the refutation process.

3.4 Examples

We now demonstrate the idea of the extensional resolution calculus on examples **E3** and **E5**:

E3 $\forall P_{o\to o}.(Pa_o \land Pb_o) \Rightarrow P(a \land b)$

CNF(¬**E3**) ($p_{o\to o}$ is a new Skolem constant):

	c1: $[pa]^T$	*c2:* $[pb]^F$	*c3:* $[p(a \land b)]^F$
Res(*c3*,*c1*):	*c4:* $[p(a \land b) = pa]^F$		
Res(*c3*,*c2*):	*c5:* $[p(a \land b) = pb]^F$		
Dec(*c4*):	*c6:* $[(a \land b) = a]^F$		
Dec(*c5*):	*c7:* $[(a \land b) = b]^F$		
Equiv(*c6*):	*c8:* $[a]^F \lor [b]^F$	*c9:* $[a]^T \lor [b]^T$	*c10:* $[a]^T$
Equiv(*c7*):	*c11:* $[a]^F \lor [b]^F$	*c12:* $[a]^T \lor [b]^T$	*c13:* $[b]^T$

The rest is obvious: Resolve *c10* and *c13* against *c8* (or *c11*). $\Box$

E5 $(\forall X_\iota.\forall P_{\iota\to o}.P(m_{\iota\to\iota}X) \Rightarrow P(n_{\iota\to\iota}X)) \Rightarrow (\forall Q_{(\iota\to\iota)\to o}.Qm \Rightarrow Qn)$

CNF(¬**E5**) (q is a new Skolem constant):

	c1: $[P(mX)]^F \lor [P(nX)]^T$	*c2:* $[qm]^T$	*c3:* $[qn]^F$
Res(*c2*,*c3*):		*c4*: $[qm = qn]^F$	
Dec(*c4*):		*c5*: $[m = n]^F$	
Func(*c5*) (s_ι is a new Skolem constant):		*c6*: $[ms = ns]^F$	
Leib(*c6*) ($p_{\iota\to o}$ is a new Skolem constant):		*c7*: $[p(ms)]^T$	*c8*: $[p(ns)]^F$

Note that resolving *c2* and *c3* immediately against *c1* does not lead to a solvable unification constraint. Instead we made a detour to the pre-unification part of the calculus and modified the clauses *c2* and *c3* in an extensionally appropriate way. Now *c2* and *c3* have their counterparts in *c7* and *c8*, but in contrast to *c2* and *c3* the new clauses can successfully be resolved against *c1*. $\Box$

The proofs of the other examples are discussed in [Ben97].

Remark 6 (Optimization of Extensionality). Note the order in which the extensionality rules were applied in the examples above. For a practical implementation these examples suggest the following **extensionality treatment** of unification constraints: First decompose the unification constraint as much as possible. Then use rule *Func* to add as many arguments as possible to both hand sides of the resulting unification constraints. And last use rule *Leib* and/or *Equiv* to finish the extensionality treatment. In this sense the above rules can be combined to form only one rule *Ext-Treat*.

Remark 7 (Rule Leib). Due to an idea of Frank Pfenning every refutation which uses rule *Leib* can possibly be done without this rule by resolving against the extensional modified unification constraint instead, and hence rule *Leib* may be superfluous. For example the application of rule *Leib* in the proof of example **E5** can be replaced by an immediate resolution step between clause *c1* and *c6*: *c7*: $[P(mX)]^F \vee [P(nX) = (ms = ns)]^F$. And by pre-unification ($P \leftarrow \lambda Y_\iota.(ms = Y)$ and $X \leftarrow s$) we immediately get the empty clause. Note that in this case it is essential that unification constraints are encoded as negative equality literals (see Remark 5).

However, there are two reasons why rule *Leib* seems to be very appropriate. First the completeness proof with respect to Henkin models seems to be more complicated without rule *Leib* and isn't done yet. Additionally the experience from the implementation work of the system LEO is, that rule *Func* eases the implementation and the integration of heuristics. See [Ben97] for a more detailed discussion.

4 Soundness and Completeness

Theorem 3 (Soundness of $\mathcal{ER}$). *The calculus $\mathcal{ER}$ is sound with respect to Henkin semantics.*

Proof. The soundness of $\mathcal{HORES}$ is discussed in detail in [Koh94], the only major difference to the first-order case is the treatment of Skolemization, which has been discussed in [Mil83].

The soundness of the three new extensionality rules are obvious, as they do only apply the two extensionality principles and the Leibniz definition, which are valid in Henkin models.

For the completeness result, we will need a series of disjunction Lemmata, which are well-known for first-order logic, and which can be proven with the same techniques, only considering the extra inference rules of $\mathcal{ER}$ in the inductions.

Lemma 1. *Let $\Phi, \Delta, \Gamma_1, \Gamma_2 \subseteq wff^{cl}(\Sigma)$ and $\mathbf{A}, \mathbf{B} \in wff^{cl}(\Sigma)$. We have*

1. *If $CNF(\Phi * \mathbf{A}) \vdash_{\mathcal{ER}} \Box$ and $CNF(\Phi * \mathbf{B}) \vdash_{\mathcal{ER}} \Box$, then $CNF(\Phi * \mathbf{A} \vee \mathbf{B}) \vdash_{\mathcal{ER}} \Box$*
2. *If $CNF(\Phi * \neg\mathbf{A} * \mathbf{B}) \vdash_{\mathcal{ER}} \Box$ and $CNF(\Phi * \mathbf{A} * \neg\mathbf{B}) \vdash_{\mathcal{ER}} \Box$, then $CNF(\Phi * \neg(\mathbf{A} \equiv \mathbf{B})) \vdash_{\mathcal{ER}} \Box$*

Proof. For the proof of the first assertion we first verify that $\mathrm{CNF}(\Phi * \mathbf{A} \vee \mathbf{B}) = \mathrm{CNF}(\Phi) \cup \mathrm{CNF}(\mathbf{A}) \sqcup \mathrm{CNF}(\mathbf{B})$, where $\Gamma \sqcup \Delta = \; := \{\mathbf{C} \vee \mathbf{D} | \mathbf{C} \in \mathrm{CNF}(A)\}, \mathbf{D} \in$

CNF(B)}. Then we use that $\Phi \cup \Gamma_1 \sqcup \Gamma_2 \vdash_{\mathcal{ER}} \Box$, provided that $\Phi \cup \Gamma_1 \vdash_{\mathcal{ER}} \Box$ and $\Phi \cup \Gamma_2 \vdash_{\mathcal{ER}} \Box$. The second involves a tedious but straightforward calculation.

Lemma 2 (Lifting Lemma). *Let Φ be a set of clauses and σ a substitution, then Φ is refutable by $\mathcal{ER}$, provided that $\theta(\Phi)$ is.*

Proof. The claim is proven by an induction on the structure of the refutation $\mathcal{D}_\theta : \theta(\Phi) \vdash_{\mathcal{ER}} \Box$ be a refutation of $\theta(\Phi)$ constructing a refutation $\mathcal{D}$ for Φ that is isomorphic to $\mathcal{D}_\theta$.

For this task it is crucial to maintain a tight correspondence $\omega : \Phi \longrightarrow \theta(\Phi)$ between the respective clause sets. This is formalized by a **clause set isomorphism**, i.e. a bijection of clause sets, that corresponding clauses are isomorphic, i.e. for a ω respects literal polarities and is compatible with θ, i.e. for any literal $\mathbf{N}^\alpha$ we have $\omega(\mathbf{N}) = \theta(\mathbf{N})$. The main difficulty with lifting properties in higher-order logic is the fact that due to the existence of predicate variables at the head of formulae, the propositional structure of formulae can change during instantiation. For instance if $\theta(F) = \lambda X_\alpha . GX \vee p$, and $\mathbf{A}^{\mathtt{T}} = Fa^{\mathtt{T}}$, then the pre-literal $\theta(F)$ is split $\mathcal{D}_\theta$ but not in the $\mathcal{ER}$-derivation already constructed. The solution of this problem is to apply the rule *Prim* with a suitable general binding $\mathcal{G}^{\vee}_{\alpha \to o} = \lambda X_\alpha . (H^1 X) \vee (H^2 X)$ and obtain a pre-literal $(H^1 a \vee H^2 a)^{\mathtt{T}}$, to which can be split in order to regain a clause set isomorphism. Since $\mathcal{G}^{\vee}_{\alpha \to o}$ is more general than $\theta(F)$ there is a substitution ρ, such that $\theta(F) = \rho(\mathcal{G}^{\vee}_{\alpha \to o})$, therefore $\omega((H^1 a \vee H^2 a)^{\mathtt{T}}) = \theta'((H^1 a \vee H^2 a)^{\mathtt{T}})$ where $\theta' = \theta \cup \rho$.

Theorem 4 (Completeness of $\mathcal{ER}$). *The calculus $\mathcal{ER}$ is complete with respect to Henkin semantics.*

Proof. Let Γ_Σ be the set of Σ-sentences which cannot be refuted by calculus $\mathcal{ER}$ ($\Gamma_\Sigma := \{\Phi \subseteq \mathit{wff}^{cl}_o(\Sigma) | \mathrm{CNF}(\Phi) \not\vdash_{\mathcal{ER}} \Box\}$), then we show that Γ_Σ is a saturated abstract consistency class for Henkin models. This entails completeness of $\mathcal{ER}$ by theorem 1.

Let $\Phi \in \Gamma_\Sigma$. We show that Φ mets the conditions required in definition 1:

∇_c Suppose that $\mathbf{A}, \neg\mathbf{A} \in \Phi$. Since $\mathbf{A}$ is atomic we have $\mathrm{CNF}(\Phi * \mathbf{A} * \neg\mathbf{A}) = \mathrm{CNF}(\Phi) * [\mathbf{A}]^T * [\mathbf{A}]^F$ and hence we can derive $\Box$ with *Res* and *Triv*. This contradicts our assumption.

In all of the remaining cases, we show the contrapositive, e.g. in the next case we prove, that for all $\Phi \in \Gamma_\Sigma$, if $\Phi * \neg\neg\mathbf{A} * \mathbf{A} \notin \Gamma_\Sigma$, then $\Phi * \neg\neg\mathbf{A} \notin \Gamma_\Sigma$, which entails the assertion.

$\nabla_\neg$ If $\mathrm{CNF}(\Phi * \neg\neg\mathbf{A} * \mathbf{A}) \vdash_{\mathcal{ER}} \Box$, then also $\mathrm{CNF}(\Phi * \neg\neg\mathbf{A}) \vdash_{\mathcal{ER}} \Box$, since $\mathrm{CNF}(\Phi * \neg\neg\mathbf{A} * \mathbf{A}) = \mathrm{CNF}(\Phi * \neg\neg\mathbf{A})$.
$\nabla_{\beta\eta}$ Analog to $\nabla_\neg$, since $\mathrm{CNF}(\Phi * \mathbf{A} * \mathbf{A}{\downarrow_{\beta\eta}}) = \mathrm{CNF}(\Phi * \mathbf{A})$.
$\nabla_\vee$ If $\mathrm{CNF}(\Phi * \mathbf{A} \vee \mathbf{B} * \mathbf{A}) \vdash_{\mathcal{ER}} \Box$ and $\mathrm{CNF}(\Phi * \mathbf{A} \vee \mathbf{B} * \mathbf{B}) \vdash_{\mathcal{ER}} \Box$, then $\mathrm{CNF}(\Phi * \mathbf{A} \vee \mathbf{B}) \vdash_{\mathcal{ER}} \Box$ by lemma 1(3).
$\nabla_\wedge$ If $\mathrm{CNF}(\Phi * \neg(\mathbf{A} \vee \mathbf{B}) * \neg\mathbf{A} * \neg\mathbf{B}) \vdash_{\mathcal{ER}} \Box$, then $\mathrm{CNF}(\Phi * \neg(\mathbf{A} \vee \mathbf{B})) \vdash_{\mathcal{ER}} \Box$, since $\mathrm{CNF}(\Phi * \neg(\mathbf{A} \vee \mathbf{B}) * \neg\mathbf{A} * \neg\mathbf{B}) = \mathrm{CNF}(\Phi * \neg(\mathbf{A} \vee \mathbf{B}))$.

$\nabla_\forall$ By the lifting lemma 2.

$\nabla_\exists$ Let $\mathrm{CNF}(\Phi * \neg\Pi\mathbf{F} * \neg\mathbf{F}w) \vdash^{D}_{\mathcal{ER}} \Box$ and note that $\mathrm{CNF}(\Phi * \neg\Pi\mathbf{F} * \neg\mathbf{F}w) = \mathrm{CNF}(\Phi * \neg\mathbf{F}w' * \neg\mathbf{F}w)$. Now let w'' be any new constant symbol which does not occur in Φ or $\mathbf{F}$. Since also w and w' do not occur in Φ or $\mathbf{F}$ it is easy to verify that their is a derivation $\mathrm{CNF}(\Phi * \neg\mathbf{F}w'') \vdash^{D'}_{\mathcal{ER}} \Box$, where each occurrence of $\neg\mathbf{F}w'$ or $\neg\mathbf{F}w$ is replaced by $\neg\mathbf{F}w''$. Hence $\mathrm{CNF}(\Phi * \neg\Pi\mathbf{F}) \vdash_{\mathcal{ER}} \Box$.

∇_b We show that if $\mathrm{CNF}(\Phi * \neg(\mathbf{A} \doteq^{o} \mathbf{B}) * \neg\mathbf{A} * \mathbf{B}) \vdash_{\mathcal{ER}} \Box$ and $\mathrm{CNF}(\Phi * \neg(\mathbf{A} \doteq^{o} \mathbf{B}) * \mathbf{A} * \neg\mathbf{B}) \vdash_{\mathcal{ER}} \Box$, then $\mathrm{CNF}(\Phi * \neg(\mathbf{A} \doteq \mathbf{B}) \vdash_{\mathcal{ER}} \Box$. Note that $\mathrm{CNF}(\Phi * \neg(\mathbf{A} \doteq \mathbf{B})) = \mathrm{CNF}(\Phi * \neg\Pi(\lambda P. \neg P\mathbf{A} \vee P\mathbf{B})) = \mathrm{CNF}(\Phi) * [r\mathbf{A}]^T * [r\mathbf{B}]^F$, with Skolem constant $r_{o\to o}$. Now consider the following derivation

$$\cfrac{\cfrac{\cfrac{[r\mathbf{A}]^T \quad [r\mathbf{B}]^F}{[r\mathbf{A} \doteq r\mathbf{B}]^F}\;\mathit{Res}}{[\mathbf{A} \doteq \mathbf{B}]^F}\;\mathit{Dec}}{\mathrm{CNF}(\neg(\mathbf{A} \equiv \mathbf{B}))}\;\mathit{Equiv}$$

Hence $\mathrm{CNF}(\Phi * \neg(\mathbf{A} \doteq \mathbf{B})) \vdash_{\mathcal{ER}} \mathrm{CNF}(\Phi * \neg(\mathbf{A} \doteq \mathbf{B})) \cup \mathrm{CNF}(\neg(\mathbf{A} \equiv \mathbf{B}))$ and we get the conclusion as a simple consequence of lemma 1(4).

∇_q We show that if $\mathrm{CNF}(\Phi * \neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}) * \neg(\mathbf{F}w \doteq^{\beta} \mathbf{G}w)) \vdash_{\mathcal{ER}} \Box$, then $\mathrm{CNF}(\Phi * \neg(\mathbf{F} \doteq \mathbf{G})) \vdash_{\mathcal{ER}} \Box$. Note that $\mathrm{CNF}(\Phi * \neg(\mathbf{F} \doteq \mathbf{G}) * \neg(\mathbf{F}w \doteq \mathbf{G}w)) = \mathrm{CNF}(\Phi * \neg\Pi(\lambda Q. \neg Q\mathbf{F} \vee Q\mathbf{G}) * \neg\Pi(\lambda P. \neg P(\mathbf{F}w) \vee P(\mathbf{G}w))) = \mathrm{CNF}(\Phi) * [q\mathbf{F}]^T * [q\mathbf{G}]^F * [p(\mathbf{F}w)]^T * [p(\mathbf{G}w)]^F$ and that $\mathrm{CNF}(\Phi * \neg(\mathbf{F} \doteq \mathbf{G})) = \mathrm{CNF}(\Phi) * [r\mathbf{F}]^T * [r\mathbf{G}]^F$, where $p_{\beta\to o}, q_{(\alpha\to\beta)\to o}$ and $r_{(\alpha\to\beta)\to o}$ are new Skolem constants. Now consider the following derivation:

$$\begin{array}{c}\cfrac{\cfrac{\cfrac{\cfrac{[r\mathbf{F}]^T \quad [r\mathbf{G}]^F}{[r\mathbf{F} \doteq r\mathbf{G}]^F}\;\mathit{Res}}{[\mathbf{F} \doteq \mathbf{G}]^F}\;\mathit{Dec}}{[\mathbf{F}s \doteq \mathbf{G}s]^F}\;\mathit{Func}}{[t(\mathbf{F}s)]^T}\;\mathit{Leib} \\ [t(\mathbf{G}s)]^F\end{array}$$

Here again s_α and $t_{\beta\to o}$ are new Skolem constants. Hence $\mathrm{CNF}(\Phi) * [r\mathbf{F}]^T * [r\mathbf{G}]^F \vdash_{\mathcal{ER}} \mathrm{CNF}(\Phi) * [r\mathbf{F}]^T * [r\mathbf{G}]^F * [t(\mathbf{F}s)]^T * [t(\mathbf{G}s)]^F$.
Now the conclusion follows from the assumption since s, t and r are only renamings of the Skolem symbols w, p and q and all do not occur in Φ.

To see that Γ_Σ is saturated let $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ and $\Phi \subseteq \mathit{wff}^{cl}_o(\Sigma)$ with $\Phi \not\vdash_{\mathcal{ER}} \Box$. We have to show that $\Phi * \mathbf{A} \not\vdash_{\mathcal{ER}} \Box$ or $\Phi * \neg\mathbf{A} \not\vdash_{\mathcal{ER}} \Box$. For that suppose $\Phi \not\vdash_{\mathcal{ER}} \Box$, but $\Phi * \mathbf{A} \vdash_{\mathcal{ER}} \Box$ and $\Phi * \neg\mathbf{A} \vdash_{\mathcal{ER}} \Box$. By lemma 1(3) we get that $\Phi * \mathbf{A} \vee \neg\mathbf{A} \vdash_{\mathcal{ER}} \Box$, and hence, since $\mathbf{A} \vee \neg\mathbf{A}$ is a tautology, it must be the case that $\Phi \vdash_{\mathcal{ER}} \Box$, which contradicts our assumption.

5 Conclusion

We have presented an extensional higher-order resolution calculus that is complete relative to Henkin model semantics. The treatment of the extensionality principles – necessary for the completeness result – by specialized (goal-directed) inference rules practical applicability, as an implentation of the calculus in the Leo-System [BK98] shows.

Acknowledgments The work reported here was funded by the Deutsche Forschungsgemeinschaft under grant HOTEL. The authors are grateful to Peter Andrews and Frank Pfenning for stimulating discussions.

References

[ABI$^+$96] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A theorem proving system for classical type theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.

[And71] Peter B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 36(3):414–432, 1971.

[And72] Peter B. Andrews. General models descriptions and choice in type theory. *Journal of Symbolic Logic*, 37(2):385–394, 1972.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Academic Press, 1986.

[And89] Peter B. Andrews. On Connections and Higher Order Logic. *Journal of Automated Reasoning*, 5:257–291, 1989.

[Bar84] H. P. Barendregt. *The Lambda Calculus.* North Holland, 1984.

[Ben97] Christoph Benzmüller. A calculus and a system architecture for extensional higher-order resolution. Research Report 97-198, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh,USA, June 1997.

[BK97] Christoph Benzmüuller and Michael Kohlhase. Model existence for higher-order logic. SEKI-Report SR-97-09, Universität des Saarlandes, 1997.

[BK98] Christoph Benzmüller and Michael Kohlhase. LEO, a higher-order theorem prover. to appear at CADE-15, 1998.

[dB72] Nicolaas Govert de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with an application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34(5):381–392, 1972.

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte der Mathematischen Physik*, 38:173–198, 1931.

[Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

[Hue73] Gérard P. Huet. A mechanization of type theory. In Donald E. Walker and Lewis Norton, editors, *Proc. IJCAI'73*, pages 139–146, 1973.

[JP76] D. C. Jensen and T. Pietrzykowski. Mechanizing ω-order type theory through unification. *Theoretical Computer Science*, 3:123–171, 1976.

[Koh94] Michael Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle.* PhD thesis, Universität des Saarlandes, 1994.

[Koh95] Michael Kohlhase. Higher-Order Tableaux. In P. Baumgartner, et al. eds, *TABLEAUX'95*, volume 918 of *LNAI*, pages 294–309, 1995.

[Mil83] Dale Miller. *Proofs in Higher-Order Logic.* PhD thesis, Carnegie-Mellon University, 1983.

[Smu63] Raymond M. Smullyan. A unifying principle for quantification theory. *Proc. Nat. Acad Sciences*, 49:828–832, 1963.