Holls: an Intentional First-Order Expression of Higher-Order Logic

Gilles Dowek, Thérèse Hardin, Claude Kirchner

HOL-$\lambda\sigma$: an intentional first-order expression of higher-order logic

Gilles Dowek, Thérèse Hardin, and Claude Kirchner

No 3556

Novembre 1998

THÈME 2

N 0249-6399

HOL-$\lambda\sigma$: an intentional first-order expression of higher-order logic

Gilles Dowek[*], Thérèse Hardin[†], and Claude Kirchner [‡]

Thème 2 — Génie logiciel
et calcul symbolique
Projet Coq, Para, Prothéo

Rapport de recherche n$^\circ$ 3556 — Novembre 1998 — 26 pages

Abstract: We give a first-order presentation of higher-order logic based on explicit substitutions. This presentation is intentionally equivalent to the usual presentation of higher-order logic based on λ-calculus, i.e. a proposition can be proved without the extensionality axioms in one theory if and only if it can in the other. We show that we can apply the *Extended Narrowing and Resolution* first-order proof-search method to this theory. We get this way a step by step simulation of higher-order resolution. Hence expressing higher-order logic as a first-order theory and applying a first-order proof search method is at least as efficient as a direct implementation. Furthermore, the well studied improvements of proof search for first-order logic could be reused at no cost for higher-order automated deduction. Moreover as we stay in a first-order setting, extensions, such as equational higher-order resolution, are easier to handle.

Key-words: higher-order logic, automated theorem proving, deduction modulo, rewriting, skolemization

(Résumé : tsvp)

[*] Gilles.Dowek@inria.fr http://coq.inria.fr/~dowek
[†] Therese.Hardin@lip6.fr http://www.lip6.fr/~hardin
[‡] Claude.Kirchner@loria.fr http://www.loria.fr/~ckirchne

HOL-$\lambda\sigma$: une expression intentionelle et au premier ordre de la logique d'ordre supérieur

Résumé : Nous proposons une présentation au premier ordre de la logique d'ordre supérieur, utilisant le calcul des substitutions explicites. Cette présentation est intentionnellement équivalente à la présentation traditionnelle de la logique d'ordre supérieur utilisant le λ-calcul : une proposition peut être démontrée sans les axiomes d'extensionnalité dans un système si et seulement si elle peut l'être dans l'autre. On montre qu'on peut appliquer à cette théorie la méthode de recherche de démonstrations pour la logique du premier ordre appelée *Surréduction et Résolution étendues*. On obtient ainsi une simulation pas à pas de la résolution d'ordre supérieur. Ainsi, exprimer la logique d'ordre supérieur comme une théorie du premier ordre et appliquer une méthode de recherche de démonstration au premier ordre est au moins aussi efficace qu'une implantation dédiée. De plus rester dans le formalisme de la logique du premier ordre permet de bénéficier des optimisations connues dans ce cadre. Enfin, cela permet d'étendre simplement la méthode, par exemple à la résolution équationnelle d'ordre supérieur.

Mots-clé : logique d'ordre supérieur, démonstration automatique, déduction modulo, réécriture, skolémisation

Introduction

Higher-order logic is a formalism that allows a natural expression of program specifications and of mathematics. It is used in many theorem provers such as HOL, Isabelle, PVS, λ-Prolog, etc. In this paper, we are concerned with automated theorem proving in this logic.

Higher-order logic can be expressed in many different ways using combinators, λ-calculus, etc. Some of these formulations express higher-order logic as a first-order theory, some other do not. Expressing higher-order logic as a first-order theory permits to use standard first-order proof-search methods. Another advantage is that extensions are easier to handle in this simple framework.

There are several ways to encode higher-order logic as a first-order theory and several proof search methods for each encoding, which are more or less efficient with respect to the standard higher-order resolution. For instance the encoding of higher-order logic using combinators is not intentionally equivalent to the standard presentation using λ-calculus, because some proofs require the extensionality axioms in this presentation, but not in the standard one. This leads to inefficiencies.

In this paper we give a first-order presentation of higher-order logic called HOL-$\lambda\sigma$ using the so called, *calculus of explicit substitutions* [ACCL91]. We show that this presentation is intentionally equivalent to the usual presentation of higher-order logic based on λ-calculus, i.e. the theories are still equivalent when we drop the extensionality axioms in both cases. We show that proof-search in this theory can be mechanized with the *Extended narrowing and resolution* (ENAR) method introduced in [DHK98]. The proof search method for higher-order logic obtained this way is as efficient as higher-order resolution and in fact simulates it step by step. It keeps however the simplicity of first-order frameworks and can easily be extended, for instance with equational axioms. A rather surprising side effect of this presentation of higher-order logic is that it provides a clarification of the intricate skolemization rule of higher-order logic [Mil83, Mil87].

The ENAR proof search method relies upon a presentation of first-order logic called *deduction modulo* that allows to build-in a congruence identifying terms and also propositions. This leads to shorter and more direct proofs by making congruent propositions equivalent instead of requiring explicit proof arguments. Hence, we shall express HOL-$\lambda\sigma$ in deduction modulo.

In order to remain self contained, we recall the principal ideas of *deduction modulo* in section 1. Then, we recall the usual presentation of higher-order logic in section 2 and its first-order presentation based on Curry combinators in section 3. Section 4 introduces HOL-$\lambda\sigma$ and etablishes its main properties (termination, confluence, consistency and cut elimination). Section 5 is dedicated to the equivalence theorem between HOL-λ and HOL-$\lambda\sigma$ (which rests upon cut elimination). In section 6 we show that the rather intricate Skolem theorem for higher-order logic can be deduced from the first-order one. At last section 7 presents the ENAR proof search method (whose completeness rests upon cut elimination) and its application to HOL-$\lambda\sigma$.

1 Deduction modulo

In this paper we shall use a presentation of first-order logic, called *deduction modulo* [DHK98] that permits to identify propositions modulo a congruence.

In deduction modulo, the notions of language, term and proposition are that of (many sorted) first-order logic. We consider theories to be formed with a set of axioms Γ *and a congruence*, denoted $\equiv$, defined on propositions. As a consequence, the deduction rules must take into account this equivalence. For instance, the *modus ponens* cannot be stated as usual

$$\frac{A \Rightarrow B \quad A}{B}$$

but, as the two occurrences of A need not be identical, but need only to be congruent, it is stated as

$$\frac{A' \Rightarrow B \quad A}{B} \text{ if } A \equiv A'$$

In fact, as the congruence may identify implications with other propositions, a slightly more general formulation is needed

$$\frac{C \quad A}{B} \text{ if } C \equiv A \Rightarrow B$$

All the rules of natural deduction or sequent calculus may be stated in a similar way. Figure 1 gives a formulation of natural deduction modulo.

As an example, the proposition $\exists x\ (2 \times x = 4)$ is rather cumbersome to prove in natural deduction with the axioms of arithmetic. Indeed to prove the proposition $2 \times 2 = 4$ we have to say that $2 \times 2 = 1 \times 2 + 2$, $1 \times 2 + 2 = 0 \times 2 + 2 + 2$, $\ldots$ and thus to use the axioms of arithmetic and equality many and many times.

In contrast, in natural deduction modulo, wehave the following proof

$$\frac{\dfrac{\dfrac{}{\forall x\ x = x}\text{ axiom}}{2 \times 2 = 4}\ (x, x = x, 4)\ \forall\text{-elim}}{\exists x\ (2 \times x = 4)}\ (x, 2 \times x = 4, 2)\ \exists\text{-intro}$$

Substituting the variable x by the term 2 in the proposition $2 \times x = 4$ yields the proposition $2 \times 2 = 4$, that is congruent to $4 = 4$. The transformation of one proposition into the other, that requires several proof steps in natural deduction, is dropped from the proof in deduction modulo. It is a mere computation that need not be written, because everybody can re-do it by him/herself.

In this case, the congruence can be defined by a rewriting system defined on terms

$$0 + y \longrightarrow y$$

$$S(x) + y \longrightarrow S(x + y)$$

$$0 \times y \longrightarrow 0$$

$$S(x) \times y \longrightarrow x \times y + y$$

Notice that, in the proof above, we do not need the axioms of addition and multiplication. Indeed, these axioms are now redundant: since the terms $0 + y$ and y are congruent, the axiom $\forall y\ 0 + y = y$ is congruent to the equality axiom $\forall y\ y = y$. Hence, it can be dropped. In other words, these axioms have been built-in [Plo72, And71, PS81].

The main originality of deduction modulo is that we have introduced the possibility to define the congruence directly on propositions with rules rewriting atomic propositions to arbitrary ones. For instance, in the theory of integral rings, we can take the rule

$$x \times y = 0 \longrightarrow x = 0 \vee y = 0$$

that rewrites an atomic proposition to a disjunction.

In this paper, all congruences will be defined by confluent rewrite systems. As these rewrite systems are defined on propositions and propositions contain binders, these rewrite systems are in fact *Combinatory reduction systems* [KvOvR93].

Notice at last, that deduction modulo is not an extension of first-order logic. Indeed, it is proved in [DHK98] that for every congruence $\equiv$, we can find a theory $\mathcal{T}$ such that $\Gamma \vdash P$ is provable modulo $\equiv$ if and only if $\mathcal{T}\Gamma \vdash P$ is provable in ordinary first-order logic. Of course, the provable propositions are the same, but the proofs are very different.

2 HOL-λ

We first recall the usual presentation of higher-order logic [Chu40, And86].

Definition 2.1 Simple types *are inductively defined by*

- *ι and o are simple types,*
- *if T and U are simple types then $T \to U$ is a simple type.*

We consider a set of typed variables and a set of typed constants, in such a way that there is an infinite set of variables of each type. Among the constants are $\dot{\Rightarrow}$, $\dot{\wedge}$ and $\dot{\vee}$, of type $o \to o \to o$, $\dot{\neg}$ of type $o \to o$, $\dot{\bot}$ of type o, $\dot{\forall}_T$ and $\dot{\exists}_T$ of type $(T \to o) \to o$ (we use a notation with a dot for the constants to distinguish them from the connectors and quantifiers of first-order logic).

Definition 2.2 Terms of type T *are inductively defined by*

- *variables and constants of type T are terms of type T,*
- *if a is a term of type $T \to U$ and b is a term of type T then $(a\ b)$ is a term of type U,*
- *if a is a term of type U and x is a variable of type T then $\lambda x\ a$ is a term of type $T \to U$.*

Propositions are terms of type o.

Definition 2.3 *(Substitution)*

- $\{b/x\}x = b$,
- $\{b/x\}y = y$,
- $\{b/x\}c = c$, *if c is a constant,*
- $\{b/x\}(c\ d) = (\{b/x\}c\ \{b/x\}d)$,
- $\{b/x\}(\lambda y\ c) = \lambda z\ (\{b/x\}\{z/y\}c)$ *where z is a fresh variable, i.e. a variable not occurring in $\lambda y\ c$ or b,*

Definition 2.4 *The $\beta\eta$-reduction is defined by the following rewrite rules*

$$((\lambda x\ t)\ u) \to \{u/x\}t$$

$$\lambda x\ (t\ x) \to t \textit{ provided } x \textit{ is not free in } t$$

$$\frac{}{\Gamma \vdash A'} \text{ axiom} \quad \text{if } A \in \Gamma, \equiv A'$$

$$\frac{\Gamma A \vdash B}{\Gamma \vdash C} \Rightarrow\text{-intro if } C \equiv (A \Rightarrow B)$$

$$\frac{\Gamma \vdash C \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow\text{-elim if } C \equiv (A \Rightarrow B)$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash C} \wedge\text{-intro if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash A} \wedge\text{-elim if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash B} \wedge\text{-elim if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash C} \vee\text{-intro if } C \equiv (A \vee B)$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash C} \vee\text{-intro if } C \equiv (A \vee B)$$

$$\frac{\Gamma \vdash D \quad \Gamma A \vdash C \quad \Gamma B \vdash C}{\Gamma \vdash C} \vee\text{-elim if } D \equiv (A \vee B)$$

$$\frac{\Gamma A \vdash \bot}{\Gamma \vdash C} \neg\text{-intro if } C \equiv \neg A$$

$$\frac{\Gamma \vdash C \quad \Gamma \vdash A}{\Gamma \vdash \bot} \neg\text{-elim if } C \equiv \neg A$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A} \bot\text{-elim if } B \equiv \bot$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash B} (x, A)\ \forall\text{-intro if } x \text{ not free in } \Gamma,\ B \equiv (\forall x\ A)$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash C} (x, A, t)\ \forall\text{-elim if } B \equiv (\forall x\ A),\ C \equiv \{t/x\}A$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash B} (x, A, t)\ \exists\text{-intro if } B \equiv (\exists x\ A),\ C \equiv \{t/x\}A$$

$$\frac{\Gamma \vdash C \quad \Gamma A \vdash B}{\Gamma \vdash B} (x, A)\ \exists\text{-elim if } x \text{ not free in } \Gamma B,\ C \equiv (\exists x\ A)$$

$$\frac{}{\Gamma \vdash B} \text{ excluded middle if } B \equiv A \vee \neg A$$

Figure 1: Natural deduction modulo

Proposition 2.1 *The $\beta\eta$-reduction is confluent and terminating. The unique normal form of a term a is written $a\downarrow$.*

The deduction rules are given in figure 2 where all propositions are supposed to be normal. An alternative presentation does not normalize the propositions after the quantifier rules but takes β and η as axioms.

This system is well-known to be consistent and to enjoy cut elimination [Gir70, Gir72].

3 HOL-C

Higher-order logic can be expressed as a many-sorted first-order theory whose sorts are all simple types. In such a presentation, when t is a term of type $T \rightarrow U$ and u a term of type T we cannot write the application of the term t to the term u as $(t\ u)$, but we need to introduce a function symbol $\alpha_{T,U}$ and write this term $\alpha_{T,U}(t,u)$. The rank of the function symbol $\alpha_{T,U}$ is $(T \rightarrow U, T)U$. Of course, in examples, we shall continue to write $(t\ u)$ for the term $\alpha_{T,U}(t,u)$.

To express function terms and predicate terms, instead of using λ-calculus, we introduce for each applicative term t whose variables are among $x_1, \ldots, x_n$ an individual symbol written $x_1, \ldots, x_n \longmapsto t$ and an axiom

$$((x_1, \ldots, x_n \longmapsto t)\ x_1\ \ldots\ x_n) = t$$

Such individual symbols are called *combinators*.

At last, we need to introduce a distinction between the terms of type o and the propositions. Hence, we introduce a predicate symbol ε of rank (o) and if t is a term of type o we write $\varepsilon(t)$ for the corresponding proposition. We introduce axioms that relate the connectors and quantifiers (e.g. $\wedge$) and their replication as individual symbols (e.g. $\dot\wedge$):

$$\varepsilon(\dot\wedge\ x\ y) \Leftrightarrow (\varepsilon(x) \wedge \varepsilon(y))$$

Thus, the language contains:

- for each applicative term t of type U whose variables are among $x_1, \ldots, x_n$ of type $T_1, \ldots, T_n$, an individual symbol $x_1, \ldots, x_n \longmapsto t$ of type $T_1 \rightarrow \ldots \rightarrow T_n \rightarrow U$,
- individual symbols $\dot\Rightarrow$, $\dot\wedge$ and $\dot\vee$ of sort $o \rightarrow o \rightarrow o$, $\dot\neg$ of type $o \rightarrow o$, $\dot\bot$ of type o, for each type T individual symbols of type $\dot\forall_T$ and $\dot\exists_T$ of type $(T \rightarrow o) \rightarrow o$,
- for each pair of types (T,U), a binary function symbol $\alpha_{T,U}$ of rank $(T \rightarrow U, T)\ U$,
- a unary predicate symbol ε of rank (o).

The axioms are

$$\varepsilon(((x_1, \ldots, x_n \longmapsto t)\ x_1\ \ldots\ x_n) = t)$$

$$\varepsilon(\dot\Rightarrow\ x\ y) \Leftrightarrow (\varepsilon(x) \Rightarrow \varepsilon(y))$$

$$\varepsilon(\dot\wedge\ x\ y) \Leftrightarrow (\varepsilon(x) \wedge \varepsilon(y))$$

$$\varepsilon(\dot\vee\ x\ y) \Leftrightarrow (\varepsilon(x) \vee \varepsilon(y))$$

$$\varepsilon(\dot\neg\ x) \Leftrightarrow \neg\varepsilon(x)$$

$$\frac{}{\Gamma \vdash A}\ \text{axiom}$$

$$\frac{\Gamma A \vdash B}{\Gamma \vdash \dot{\Rightarrow}\ A\ B}\ \dot{\Rightarrow}\text{-intro}$$

$$\frac{\Gamma \vdash \dot{\Rightarrow}\ A\ B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \dot{\Rightarrow}\text{-elim}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash \dot{\wedge}\ A\ B}\ \dot{\wedge}\text{-intro}$$

$$\frac{\Gamma \vdash \dot{\wedge}\ A\ B}{\Gamma \vdash A}\ \dot{\wedge}\text{-elim}$$

$$\frac{\Gamma \vdash \dot{\wedge}\ A\ B}{\Gamma \vdash B}\ \dot{\wedge}\text{-elim}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \dot{\vee}\ A\ B}\ \dot{\vee}\text{-intro}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash \dot{\vee}\ A\ B}\ \dot{\vee}\text{-intro}$$

$$\frac{\Gamma \vdash \dot{\vee}\ A\ B \quad \Gamma A \vdash C \quad \Gamma B \vdash C}{\Gamma \vdash C}\ \dot{\vee}\text{-elim}$$

$$\frac{\Gamma A \vdash \dot{\bot}}{\Gamma \vdash \dot{\neg} A}\ \dot{\neg}\text{-intro}$$

$$\frac{\Gamma \vdash \dot{\neg} A \quad \Gamma \vdash A}{\Gamma \vdash \dot{\bot}}\ \dot{\neg}\text{-elim}$$

$$\frac{\Gamma \vdash \dot{\bot}}{\Gamma \vdash A}\ \dot{\bot}\text{-elim}$$

$$\frac{\Gamma \vdash (A\ x) \downarrow}{\Gamma \vdash \dot{\forall} A}\ \dot{\forall}\text{-intro if } x \text{ not free in } \Gamma$$

$$\frac{\Gamma \vdash \dot{\forall}\ A}{\Gamma \vdash (A\ t) \downarrow}\ \dot{\forall}\text{-elim}$$

$$\frac{\Gamma \vdash (A\ t) \downarrow}{\Gamma \vdash \dot{\exists}\ A}\ \dot{\exists}\text{-intro}$$

$$\frac{\Gamma \vdash \dot{\exists}\ A \quad \Gamma (A\ x) \downarrow \vdash B}{\Gamma \vdash B}\ \dot{\exists}\text{-elim if } x \text{ not free in } \Gamma B$$

$$\frac{}{\Gamma \vdash A \dot{\vee} \dot{\neg} A}\ \text{excluded middle}$$

Figure 2: **HOL-$\boldsymbol{\lambda}$**: The deduction rules of HOL-λ

$$\varepsilon(\dot{\bot}) \Leftrightarrow \bot$$
$$\varepsilon(\dot{\forall}\ x) \Leftrightarrow \forall y\ \varepsilon(x\ y)$$
$$\varepsilon(\dot{\exists}\ x) \Leftrightarrow \exists y\ \varepsilon(x\ y)$$

These axioms can be dropped if we work modulo the congruence defined by the rewrite system $\mathcal{R}$:

$$((x_1, \dots, x_n \longmapsto t)\ x_1\ \dots\ x_n) \longrightarrow t$$
$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow \varepsilon(x) \Rightarrow \varepsilon(y)$$
$$\varepsilon(\dot{\wedge}\ x\ y) \longrightarrow \varepsilon(x) \wedge \varepsilon(y)$$
$$\varepsilon(\dot{\vee}\ x\ y) \longrightarrow \varepsilon(x) \vee \varepsilon(y)$$
$$\varepsilon(\dot{\neg}\ x) \longrightarrow \neg\varepsilon(x)$$
$$\varepsilon(\dot{\bot}) \longrightarrow \bot$$
$$\varepsilon(\dot{\forall}\ x) \longrightarrow \forall y\ \varepsilon(x\ y)$$
$$\varepsilon(\dot{\exists}\ x) \longrightarrow \exists y\ \varepsilon(x\ y)$$

Translation from λ-terms to combinators is usualy called λ-*lifting*. A term of the form $\lambda x\ t$ is translated as follows. We first translate t yielding a term t'. We let $y_1, \dots, y_n$ be the variables of t' minus x, then we replace in t' all the occurrences of a combinator c_i by a fresh variable z_i yielding a term t''. The term $\lambda x\ t$ is then translated as the term $((y_1, \dots, y_n, z_1, \dots, z_p, x \longmapsto t'')\ y_1\ \dots\ y_n\ c_1\ \dots\ c_p)$. Applications, variables and constants are translated in an obvious way. This translation can be modified in order to use only two combinators $S = x, y, z \longmapsto ((x\ z)\ (y\ z))$ and $K = x, y \longmapsto x$.

This presentation of higher-order logic can be shown to be equivalent to the presentation with λ-calculus if we take the extensionality axioms

$$\forall f\ \forall g\ ((\forall x\ (fx) = (gx)) \Rightarrow f = g)$$

$$\forall x\ \forall y\ (\varepsilon(x \dot{\Leftrightarrow} y) \Rightarrow x = y)^{1}$$

in both cases, i.e. a proposition P is provable in the presentation of higher-order logic with λ-calculus if and only if the proposition $\varepsilon(P')$ is provable in the first-order theory above.

But, if we drop the extensionality axioms, then the two presentations are not equivalent anymore. For instance, the proposition

$$((\lambda x\ \lambda y\ x)\ (u\ v)) = \lambda y\ (u\ v)$$

is provable in presentation with λ-calculus while its translation

$$((f, x \longmapsto (f\ x))\ (x, y \longmapsto x)\ (u\ v)) = ((u, v, y \longmapsto (u\ v))\ u\ v)$$

requires extensionality. Even when the extensionality axioms are taken, the formulations with λ-calculus and combinators are only weakly equivalent: provable propositions are the same, but the proofs are very different, some proofs requiring only conversion when expressed with λ-calculus and the use of extensionality when expressed with combinators.

[1] Here, equality is Leibniz' equality, i.e. $\lambda x\ \lambda y\ \dot{\forall}\lambda p\ ((p\ x)\dot{\Rightarrow}(p\ y))$

4 HOL-$\lambda\sigma$

In order to stay in a first-order setting but to avoid the previously mentioned drawbacks of combinators, we present in this section another first-order formulation of higher-order logic that is not based on combinators, but on de Bruijn indices and explicit substitutions.

4.1 The theory

In λ-calculus with de Bruijn indices, bound variables are replaced by an index indicating the binding height of this variable, i.e. the number of λ's between this occurrence and its binder. For instance the term $\lambda x\ (x\ (\lambda y\ x))$ is written $\lambda\ (1\ (\lambda\ 2))$. This notation is also a first-order language with a binary function symbol α, a unary function symbol λ and individual symbols $1, 2, 3 \ldots$. Simple sorts are not sufficient anymore with de Bruijn indices. Indeed, we need to give a sort not only to terms like $(\lambda_A\ 1)$ (that gets the sort $A \to A$), but also to terms of the form 1. Thus, as detailed in [DHK95], we have to consider sorts of the form $\Gamma \vdash T$ where T is a simple type and Γ a context, i.e. a list of simple types.

With de Bruijn indices conversion axioms use an external definition for substitution. Moreover this substitution is not well-defined on open terms of this first-order language. This is solved by considering an extension of this calculus: the calculus of explicit substitutions [ACCL91] called $\lambda\sigma$-calculus. This calculus introduced more sorts of the form $\Gamma \vdash \Delta$ for substitutions that are lists of terms and symbols to build such substitutions id, ., $\uparrow$ and $\circ$. Then a new term constructor is introduced $\_[\_]$ that permits to apply an explicit substitution to a term. The rewrite rules describing the evaluation of the $\lambda\sigma$-calculus are given in figure 3.

HOL-$\lambda\sigma$ is a many-sorted first-order theory with sorts of the form $\Gamma \vdash T$ and $\Gamma \vdash \Delta$ where Γ and Δ are sequences of simple types and T is a simple type.

Definition 4.1 *(Language) The language contains the following function symbols:*

1^{Γ}_{A}	*constant of sort*	$A.\Gamma \vdash A$
$\alpha^{\Gamma}_{A\to B,A}$	*binary function of rank*	$(\Gamma \vdash A \to B, \Gamma \vdash A)\Gamma \vdash B$
$\lambda^{\Gamma}_{A,B}$	*unary function of rank*	$(A.\Gamma \vdash B)\Gamma \vdash A \to B$
$[\ ]^{\Gamma,\Gamma'}_{A}$	*binary function of rank*	$(\Gamma' \vdash A, \Gamma \vdash \Gamma')\Gamma \vdash A$
id^{Γ}	*constant of sort*	$\Gamma \vdash \Gamma$
$\uparrow^{\Gamma}_{A}$	*constant of sort*	$A.\Gamma \vdash \Gamma$
$\cdot^{\Gamma,\Gamma'}_{A}$	*binary function of rank*	$(\Gamma \vdash A, \Gamma \vdash \Gamma')\Gamma \vdash A.\Gamma'$
$\circ^{\Gamma,\Gamma',\Gamma''}$	*binary function of rank*	$(\Gamma \vdash \Gamma'', \Gamma'' \vdash \Gamma')\Gamma \vdash \Gamma'$
$\dot{\Rightarrow}$	*constant of sort*	$\vdash o \to o \to o$
$\dot{\wedge}$	*constant of sort*	$\vdash o \to o \to o$
$\dot{\vee}$	*constant of sort*	$\vdash o \to o \to o$
$\dot{\neg}$	*constant of sort*	$\vdash o \to o$
$\dot{\bot}$	*constant of sort*	$\vdash o$
$\dot{\forall}_A$	*constant of sort*	$\vdash (A \to o) \to o$
$\dot{\exists}_A$	*constant of sort*	$\vdash (A \to o) \to o$

and a single unary predicate symbol:

β-reduction and η-reduction:

$$(\lambda a)b \longrightarrow a[b.id]$$

$$\lambda(a\ 1) \longrightarrow b \text{ if } a =_\sigma b[\uparrow]$$

σ-reduction:

$$(a\ b)[s] \longrightarrow (a[s]\ b[s])$$

$$\mathtt{1}[a.s] \longrightarrow a$$

$$a[id] \longrightarrow a$$

$$(\lambda a)[s] \longrightarrow \lambda(a[\mathtt{1}.(s \circ \uparrow)])$$

$$(a[s])[t] \longrightarrow a[s \circ t]$$

$$id \circ s \longrightarrow s$$

$$\uparrow \circ\ (a.s) \longrightarrow s$$

$$(s_1 \circ s_2) \circ s_3 \longrightarrow s_1 \circ (s_2 \circ s_3)$$

$$(a.s) \circ t \longrightarrow a[t].(s \circ t)$$

$$s \circ id \longrightarrow s$$

$$\mathtt{1}. \uparrow \longrightarrow id$$

$$\mathtt{1}[s].(\uparrow \circ\ s) \longrightarrow s$$

Figure 3: The rewrite rules of $\lambda\sigma$-calculus

$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow \varepsilon(x) \Rightarrow \varepsilon(y)$$
$$\varepsilon(\dot{\wedge}\ x\ y) \longrightarrow \varepsilon(x) \wedge \varepsilon(y)$$
$$\varepsilon(\dot{\vee}\ x\ y) \longrightarrow \varepsilon(x) \vee \varepsilon(y)$$
$$\varepsilon(\dot{\neg}\ x) \longrightarrow \neg\varepsilon(x)$$
$$\varepsilon(\dot{\bot}) \longrightarrow \bot$$
$$\varepsilon(\dot{\forall}_T\ x) \longrightarrow \forall y\ \varepsilon(x\ y)$$
$$\varepsilon(\dot{\exists}_T\ x) \longrightarrow \exists y\ \varepsilon(x\ y)$$

Figure 4: The $\mathcal{L}$-rewrite rules

$$\varepsilon \quad \textit{of rank} \quad (\vdash o)$$

We denote $\lambda\sigma\mathcal{L}$ the rewrite rules of $\lambda\sigma$-calculus together with the logical rules $\mathcal{L}$ given in figure 4 and we write $A \equiv B$ when A and B are congruent modulo $\lambda\sigma\mathcal{L}$.

4.2 Termination and confluence

Proposition 4.1 *(Termination) The system $\lambda\sigma\mathcal{L}$ is weakly terminating.*

Proof. Since the $\mathcal{L}$ and $\lambda\sigma$ rewrite systems share the application operator α, we cannot try to apply the existing termination modularity results.

We define a translation of the terms and the propositions of HOL-$\lambda\sigma$ into the typed system $\lambda\sigma$. In each sort $\Gamma \vdash T$, we chose a variable $z_{\Gamma\vdash T}$.

- $||x|| = z_T$, where x is a variable,
- every symbol is mapped to itself but:
  - $||\dot{\Rightarrow}|| = ||\dot{\wedge}|| = ||\dot{\vee}|| = ((\lambda 1)\ z_{\vdash o\to o\to o})$,
  - $||\dot{\neg}|| = (\lambda 1)$,
  - $||\dot{\bot}|| = ((\lambda 1)\ z_{\vdash o})$,
  - $||\dot{\forall}_T|| = ||\dot{\exists}_T|| = \lambda(1\ z_{\vdash T}[\uparrow])$,
- $||\varepsilon(t)|| = ||t||$,
- $||P \Rightarrow Q|| = ||P \wedge Q|| = ||P \vee Q|| = (z_{\vdash o\to o\to o}\ ||P||\ ||Q||)$,
- $||\neg P|| = ||P||$,
- $||\bot|| = z_o$,
- $||\forall x\ P|| = ||\exists x\ P|| = ||P||$.

In $\lambda\sigma\mathcal{L}$, we say that t R_1-reduces to u if u is obtained by reducing a β-redex, a η-redex or a logical redex and σ-normalizing the term obtained. In $\lambda\sigma$, we say that t R_2-reduces to u if u is obtained by reducing a β-redex or a η-redex and σ-normalizing the term obtained. We check that if P R_1-rewrites in one step to Q, then $||P||$ R_2-rewrites in at least one step to $||Q||$. Let $P_1, P_2, \ldots$ be a R_1-reduction sequence in the system above, the sequence $||P_1||, ||P_2||, \ldots$ is a R_2-reduction sequence in $\lambda\sigma$, thus it is finite [GL97, Muñ97]. □

Proposition 4.2 *(Confluence) This system is confluent on terms containing only term variables.*

Proof. Since the $\mathcal{L}$ and $\lambda\sigma$ rewrite systems share the application operator α, we cannot apply Toyama's modularity result.

The system $\mathcal{L}$ is linear and orthogonal, hence it is strongly confluent (i.e. if $t \to^1_{\mathcal{L}} u$ and $t \to^1_{\mathcal{L}} v$ then there exists a term w such that $u \to^1_{\mathcal{L}} w$ and $v \to^1_{\mathcal{L}} w$). Since $\lambda\sigma$ is confluent [ACCL91], the rewrite system $\lambda\sigma^*$ is strongly confluent. At last $\mathcal{L}$ and $\lambda\sigma^*$ strongly commute (i.e. if $t \to^1_{\mathcal{L}} u$ and $t \to^*_{\lambda\sigma} v$ then there exists a term w such that $u \to^*_{\lambda\sigma} w$ and $v \to^1_{\mathcal{L}} w$). Indeed if $t \to^1_{\mathcal{L}} u$ and $t \to^*_{\lambda\sigma} v$ then the $\mathcal{L}$-redex in t is either disjoint from or higher than the $\lambda\sigma^*$ redex. In both cases we can reduce the $\lambda\sigma$-redex in u and the $\mathcal{L}$ redex in v getting the same term. Hence, by Hindley-Rosen lemma [Hin64], the relation $\to_{\mathcal{L}} \cup (\to_{\lambda\sigma^*})$, i.e. $\lambda\sigma\mathcal{L}$ is confluent. □

4.3 Consistency

Proposition 4.3 *(Consistency) The theory HOL-λσ is consistent.*

Proof. We construct a model as follows:

- $\mathcal{M}_\iota = \{0\}$,
- $\mathcal{M}_o = \{0, 1\}$,
- $\mathcal{M}_{T \to U} = \mathcal{M}_U^{\mathcal{M}_T}$,
- $\mathcal{D}_{T_1,\ldots,T_n \vdash U} = \mathcal{M}_U{}^{\mathcal{M}_{T_n}{}^{\cdots^{\mathcal{M}_{T_1}}}}$,
- $\mathcal{D}_{T_1,\ldots,T_n \vdash U_1,\ldots,U_p} = (\mathcal{M}_{U_1} \times \ldots \times \mathcal{M}_{U_p})^{\mathcal{M}_{T_n}{}^{\cdots^{\mathcal{M}_{T_1}}}}$.

If f is a function of $(\mathcal{M}_{U_1} \times \ldots \times \mathcal{M}_{U_p})^{\mathcal{M}_{T_n}{}^{\cdots^{\mathcal{M}_{T_1}}}}$ and g a function of $(\mathcal{M}_{V_1} \times \ldots \times \mathcal{M}_{V_q})^{\mathcal{M}_{U_p}{}^{\cdots^{\mathcal{M}_{U_1}}}}$ we write $g \circ f$ for the function of $(\mathcal{M}_{V_1} \times \ldots \times \mathcal{M}_{V_q})^{\mathcal{M}_{T_n}{}^{\cdots^{\mathcal{M}_{T_1}}}}$ mapping $x_1, \ldots, x_n$ to $g(t_1)\ldots(t_p)$ where $(t_1, \ldots, t_p) = f(x_1)\ldots(x_n)$

Then we interpret the symbols of the language as follows.

- $\overline{\mathbf{1}^\Gamma_A}$ is the function mapping $a_1, \ldots, a_n$ to a_1.
- $\overline{\alpha^\Gamma_{A\to B,A}}$ is the function mapping a and b to the function mapping $c_1, \ldots, c_n$ to $a(c_1, \ldots, c_n)(b(c_1, \ldots, c_n))$.
- $\overline{\lambda^\Gamma_{A,B}}$ is the identity function.

- $\overline{[\;]_A^{\Gamma,\Gamma'}}$ is the function mapping a and b to $b \circ a$.
- $\overline{id^{\Gamma}}$ is the identity function.
- $\overline{\uparrow_A^{\Gamma}}$ is the function mapping $a_1, \dots, a_n$ to $(a_2, \dots, a_n)$.
- $\overline{\cdot_A^{\Gamma,\Gamma'}}$ is the function mapping a, b to the function mapping $c_1, \dots, c_n$ to $(a(c_1, \dots, c_n), b(c_1, \dots, c_n))$.
- $\overline{\circ^{\Gamma,\Gamma',\Gamma''}}$ is the function mapping a and b to $b \circ a$.
- $\overline{\dot{\Rightarrow}}$ is the function mapping a and b to 1 if $a = 0$ or $b = 1$ and to 0 otherwise.
- $\overline{\dot{\wedge}}$ is the function mapping a and b to 1 if $a = 1$ and $b = 1$ and to 0 otherwise.
- $\overline{\dot{\vee}}$ is the function mapping a and b to 1 if $a = 1$ or $b = 1$ and to 0 otherwise.
- $\overline{\dot{\neg}}$ is the function mapping a to 1 if $a = 0$ and to 0 otherwise.
- $\overline{\dot{\bot}} = 0$.
- $\overline{\dot{\forall}_A}$ is the function mapping a to 1 if a maps every object of $\mathcal{M}_A$ to 1 and to 0 otherwise.
- $\overline{\dot{\exists}_A}$ is the function mapping a to 1 if a maps some object of $\mathcal{M}_A$ to 1 and to 0 otherwise.
- $\overline{\varepsilon}$ is the identity function.

We check that if $A \equiv B$ then A and B have the same denotation. Then we check that every provable proposition has for denotation the truth value 1 and hence that $\bot$ is not derivable. □

4.4 Cut elimination

Proposition 4.4 *(Proof normalization) Proofs modulo $\lambda\sigma\mathcal{L}$ normalize.*

Proof. Following the method developed in [DW98] we construct a pre-model of the rewrite system above.

We recall that *proofs-terms* are inductively defined as follows.

$$\begin{array}{ll} \pi ::= & \alpha \\ & \mid \lambda\alpha\ \pi \mid (\pi\ \pi') \\ & \mid (\pi, \pi') \mid fst(\pi) \mid snd(\pi) \\ & \mid i(\pi) \mid j(\pi) \mid (\delta\ \pi_1\ \alpha\pi_2\ \beta\pi_3) \\ & \mid (botelim\ \pi) \\ & \mid \lambda x\ \pi \mid (\pi\ t) \\ & \mid (t, \pi) \mid (exelim\ \pi\ x\alpha\pi') \end{array}$$

And on these proof terms reduction rules are

$$(\lambda\alpha\ \pi_1\ \pi_2) \rhd [\pi_2/\alpha]\pi_1$$

$$fst(\pi_1, \pi_2) \rhd \pi_1$$

$$snd(\pi_1, \pi_2) \rhd \pi_2$$
$$(\delta\ i(\pi_1), \alpha\pi_2, \beta\pi_3) \rhd [\pi_1/\alpha]\pi_2$$
$$(\delta\ j(\pi_1), \alpha\pi_2, \beta\pi_3) \rhd [\pi_1/\beta]\pi_3$$
$$(\lambda x\ \pi\ t) \rhd [t/x]\pi$$
$$(exelim\ (t, \pi_1)\ \alpha x\pi_2) \rhd [t/x, \pi_1/\alpha]\pi_2$$

$$(\delta\ \pi_1\ \alpha\pi_2\ \beta\pi_3) \rhd \pi_2$$
$$(\delta\ \pi_1\ \alpha\pi_2\ \beta\pi_3) \rhd \pi_3$$
$$(exelim\ \pi_1\ x\alpha\pi_2) \rhd \pi_2$$

We recall that a proof term is said to be *neutral* if it is a proof variable or an elimination (i.e. of the form $(\pi\ \pi')$, $fst(\pi)$, $snd(\pi)$, $(\delta\ \pi_1\ \alpha\pi_2\ \beta\pi_3)$, $(botelim\ \pi)$, $(\pi\ t)$, $(exelim\ \pi\ x\alpha\pi')$, but not an introduction) and that a set R of proofs is a *reducibility candidate* if

- if $\pi \in R$, then π is strongly normalizable,
- if $\pi \in R$ and $\pi \rhd \pi'$ then $\pi' \in R$,
- if π is neutral and if for every π' such that $\pi \rhd^1 \pi'$, $\pi' \in R$ then $\pi \in R$.

We write $\mathcal{C}$ for the set of all reducibility candidates. A *pre-model* for a language $\mathcal{L}$ is given by:

- for each sort T a set $\mathcal{M}_T$,
- for each function symbol f (of rank $(T_1, \dots, T_n, U)$) a function $\overline{f}$ of $\mathcal{M}_U^{\mathcal{M}_{T_1} \times \dots \times \mathcal{M}_{T_n}}$,
- for each predicate symbol P (of rank $(T_1, \dots, T_n)$) a function $\overline{P}$ of $\mathcal{C}^{\mathcal{M}_{T_1} \times \dots \times \mathcal{M}_{T_n}}$.

Let t be a term and φ an assignment mapping all the free variables of t of sort T to elements of $\mathcal{M}_T$. We define the object $|t|_\varphi$ by induction over the structure of t.

- $|x|_\varphi = \varphi(x)$,
- $|f(t_1, \dots, t_n)|_\varphi = \overline{f}(|t_1|_\varphi, \dots, |t_n|_\varphi)$.

Let A be a proposition and φ an assignment mapping all the free variables of A of sort T to elements of $\mathcal{M}_T$. We define the set $|A|_\varphi$ of proofs by induction over the structure of A.

- A proof π is an element of $|P(t_1, \dots, t_n)|_\varphi$ if it is an element of $\overline{P}(|t_1|_\varphi, \dots, |t_n|_\varphi)$.
- A proof π is element of $|A \Rightarrow B|_\varphi$ if it is strongly normalizable and when π reduces to a proof of the form $\lambda\alpha\pi_1$ then for every π' in $|A|_\varphi$, $[\pi'/\alpha]\pi_1$ is an element of $|B|_\varphi$.
- A proof π is an element of $|A \wedge B|_\varphi$ if it is strongly normalizable and when π reduces to a proof of the form (π_1, π_2) then π_1 and π_2 are elements of $|A|_\varphi$ and $|B|_\varphi$.
- A proof π is an element of $|A \vee B|_\varphi$ if it is strongly normalizable and when π reduces to a proof of the form $i(\pi_1)$ (resp. $j(\pi_2)$) then π_1 (resp. π_2) is an element of $|A|_\varphi$ (resp. $|B|_\varphi$).

- A proof π is an element of $|\bot|_\varphi$ if it is strongly normalizable.
- A proof π is an element of $|\forall x\ A|_\varphi$ if it is strongly normalizable and when π reduces to a proof of the form $\lambda x\ \pi_1$ then for every term t of sort T (where T is the sort of x) and every element E of $\mathcal{M}_T$ $[t/x]\pi_1$ is an element of $|A|_{\varphi+(x,E)}$.
- A proof π is an element of $|\exists x\ A|_\varphi$ if it is strongly normalizable and when π reduces to a proof of the form (t, π_1) then for every element E of $\mathcal{M}_T$ (where T is the sort of t) then π_1 is an element of $|A|_{\varphi+(x,E)}$.

A pre-model is a pre-model of $\equiv$ if when $A \equiv B$ then for every assignment φ, $|A|_\varphi = |B|_\varphi$.

It is proved in [DW98] that the cut elimination property holds for intuitionistic natural deduction and sequent calculus modulo a congruence if we can build a pre-model of this congruence.

We let

- $\mathcal{M}_\iota = \{0\}$,
- $\mathcal{M}_o = \mathcal{C}$, i.e. the set of all reducibility candidates.
- $\mathcal{M}_{T\to U} = \mathcal{M}_U^{\mathcal{M}_T}$.
- $\mathcal{D}_{T_1,\dots,T_n\vdash U} = {\mathcal{M}_U}^{{\mathcal{M}_{T_n}}^{\dots^{\mathcal{M}_{T_1}}}}$,
- $\mathcal{D}_{T_1,\dots,T_n\vdash U_1,\dots,U_p} = (\mathcal{M}_{U_1} \times \ldots \times \mathcal{M}_{U_p})^{{\mathcal{M}_{T_n}}^{\dots^{\mathcal{M}_{T_1}}}}$.

Then we interpret the symbols of the language as follows.

- $\overline{1^\Gamma_A}$ is the function mapping $a_1, \dots, a_n$ to a_1.
- $\overline{\alpha^\Gamma_{A\to B,A}}$ is the function mapping a and b to the function mapping $c_1, \dots, c_n$ to $a(c_1, \dots, c_n)(b(c_1, \dots, c_n))$.
- $\overline{\lambda^\Gamma_{A,B}}$ is the identity function.
- $\overline{[]^{\Gamma,\Gamma'}_A}$ is the function mapping a and b to $b \circ a$.
- $\overline{id^\Gamma}$ is the identity function.
- $\overline{\uparrow^\Gamma_A}$ is the function mapping $a_1, \dots, a_n$ to $(a_2, \dots, a_n)$
- $\overline{\cdot^{\Gamma,\Gamma'}_A}$ is the function mapping a, b to the function mapping $c_1, \dots, c_n$ to $(a(c_1, \dots, c_n), b(c_1, \dots, c_n))$.
- $\overline{\circ^{\Gamma,\Gamma',\Gamma''}}$ is the function mapping a and b to $b \circ a$.
- $\overline{\dot{\Rightarrow}}$ is the function mapping a and b to the set of proofs π such that π is strongly normalizable and when π reduces to a proof of the form $\lambda\alpha\pi_1$ then for every π' in a, $\{\pi'/\alpha\}\pi_1$ is an element of b.
- $\overline{\dot{\wedge}}$ is the function mapping a and b to the set of proofs π such that π is strongly normalizable and when π reduces to a proof of the form (π_1, π_2) then π_1 is an element of a and π_2 is an element of b.

- $\overline{\dot{\vee}}$ is the function mapping a and b to the set of proofs π such that π is strongly normalizable and when π reduces to a proof of the form $i(\pi_1)$ then π_1 is an element of a and when π reduces to a proof of the form $j(\pi_2)$ then π_2 is an element of b.
- $\overline{\dot{\neg}}$ is the function mapping a to the set of proofs π such that π is strongly normalizable and when π reduces to a proof of the form $\lambda\alpha\pi_1$ then for every π' in a, $\{\pi'/\alpha\}\pi_1$ is strongly normalizable.
- $\overline{\dot{\bot}}$ if the set of strongly normalizable proofs.
- $\overline{\dot{\forall}}$ is the function mapping a to the set of proofs π such that π is strongly normalizable and when π reduces to a proof of the form $\lambda x\ \pi_1$ then for every term t of sort T (where T is the sort of x) and every element E of $\mathcal{M}_T$ $\{t/x\}\pi_1$ is an element of $(a\ E)$,
- $\overline{\dot{\exists}_A}$ is the function mapping a to the set of proofs π such that π is strongly normalizable and when π reduces to a proof of the form (t, π_1) then for every element E of $\mathcal{M}_T$ (where T is the sort of x) $\{t/x\}\pi_1$ is an element of $(a\ E)$.
- $\overline{\varepsilon}$ is the identity function.

And we check that the rewrite rules are valid in this pre-model, hence the rewrite system has a pre-model and proof modulo this rewrite system normalize. □

Following the techniques introduced in [DW98] we can prove also that the classical sequent calculus has the cut elimination property.

5 Embedding HOL-λ into HOL-$\lambda\sigma$

We now want to prove that HOL-$\lambda\sigma$ is intentionally equivalent to the usual presentation of higher-order logic HOL-λ.

Following [DHK95], we define a translation from λ-calculus to $\lambda\sigma$-calculus called *pre-cooking*. This translation replaces the bound variables by the appropriate indices and adds an appropriate $[\uparrow^n]$ operator to free variables and constants according to the context in which they occur.

To each variable x of type T, we associate the sort $\vdash T$ in $\lambda\sigma$-calculus.

Definition 5.1 *Let a be a λ-term. The pre-cooking of a is the $\lambda\sigma$-term defined by $a_F = F(a, [\ ])$ where $F(a, l)$ is defined using the list of variable l ($[\ ]$ being the empty list) by:*

- $F((\lambda x.a), l) = \lambda(F(a, x.l))$,
- $F((a\ b), l) = F(a, l)F(b, l)$,
- $F(x, l) = 1[\uparrow^{k-1}]$, *if x is the k-th variable of l*
- $F(x, l) = x[\uparrow^n]$ *where n is the length of l if x is a variable not occurring in l or a constant.*

Theorem 5.1 *If $p_1, \ldots, p_n, q$ are propositions in HOL-λ then $p_1, \ldots, p_n \vdash q$ is provable in HOL-λ if and only if $\varepsilon(p_{1F}), \ldots, \varepsilon(p_{nF}) \vdash \varepsilon(q_F)$ is provable in HOL-$\lambda\sigma$.*

The proof of this result relies on the propositions that we are now introducing and proving.

Proposition 5.1 *If t has the type T then t_F has the sort $\vdash T$.*

Proof. By induction on the structure of t. $\square$

Proposition 5.2

- $(\{a/x\}b)_F = \{x \mapsto a_F\}b_F$,
- *$a =_{\beta\eta} b$ in λ-calculus if and only if $a_F =_{\lambda\sigma} b_F$ in $\lambda\sigma$-calculus.*

Proof. See [DHK95]. $\square$

The purpose of the following definition and proposition is to characterize the image of the pre-cooking mapping.

Definition 5.2 *A F-*term *is a $\lambda\sigma$-term containing only variables which sort has an empty context. A F-*proposition *is a proposition of the form $\varepsilon(P)$ where P is a F-term.*

Proposition 5.3 *If t is a $\lambda\sigma\mathcal{L}$-normal F-term well-typed in the empty context then there is a λ-term u such that $t = u_F$.*

Proof. We prove by induction on the structure of t that if t is a $\lambda\sigma\mathcal{L}$-normal F-term well-typed in a context Γ, then there is a term u and a sequence l of variables of the same length than Γ such that $t = F(u, l)$.

The only interesting case is when $t = x[s]$. This term is well-typed in a context Γ of length n thus s has type $\Gamma \vdash$ and it is normal, thus $s =\uparrow^n$. $\square$

Proposition 5.4 *If T is a set of F-propositions and P a F-proposition, and the proposition P has a proof under the assumptions T, then it also has a proof where all propositions are F-propositions and all the witnesses F-terms.*

Proof. By induction on the size of a cut free proof of $T \vdash P$.

- If the last rule is an axiom then the result is obvious.
- If the last rule is an introduction rule, we apply the induction hypothesis to the subproofs. The only non trivial case is the introduction of the existential quantifier. The proof has the form

$$\dfrac{\dfrac{\pi}{T \vdash R}}{T \vdash \varepsilon(q)}\ (x, P, t)\ \exists\text{-intro}$$

Where $\varepsilon(q) \equiv \exists x\ P$ and $R \equiv \{t/x\}P$. Hence $q \equiv (\dot{\exists}\ p)$, $P \equiv \varepsilon(p\ x)$ and $R \equiv \varepsilon(p\ t)$. Call σ the substitution mapping each variable x of t of sort $A_1, \ldots , A_n \vdash B$ to the term $x'[\uparrow^n]$ where x' is a fresh variable of sort $\vdash B$. By induction on the structure of π, the proof $\sigma\pi$ is a proof of $T \vdash \sigma R$, i.e. $T \vdash \varepsilon(p\ \sigma t)$. We apply the induction hypothesis to the proof $\sigma\pi$. Hence, there is a proof π' of $T' \vdash \varepsilon(p\ \sigma t)$ where all propositions are F-propositions and all the witnesses F-terms.

We build the proof

$$\dfrac{\dfrac{\pi'}{T \vdash \varepsilon(p\ \sigma t)}}{T \vdash \varepsilon(q)}\ (x, \varepsilon(p\ x), \sigma t)\ \exists\text{-intro}$$

- If the proof ends with an elimination rule, then it is a non empty sequence of eliminations, followed by an axiom on a proposition $\varepsilon(p)$ of T. We reason by cases on the first elimination rule after this axiom. We apply the induction hypothesis to the subproofs.

  For instance, if this rule is an elimination of an implication, the proof has the form

$$\dfrac{\dfrac{\dfrac{}{T \vdash \varepsilon(p)}\text{axiom} \quad \dfrac{\rho}{T \vdash Q}}{T \vdash R}}{\dfrac{\pi}{T \vdash \varepsilon(s)}}$$

  where $\varepsilon(p) \equiv Q \Rightarrow R$. Hence, $p \equiv (\dot{\Rightarrow}\ q\ r)$, $Q \equiv \varepsilon(q)$, and $R \equiv \varepsilon(r)$.

  Form the proof π we can build a proof π_1 of $T\varepsilon(r) \vdash \varepsilon(s)$ by adding the proposition $\varepsilon(r)$ to all the contexts and adding an axiom rule on the top. We apply the induction hypothesis to the proofs ρ and π_1. Hence we have proof ρ' and π'_1 of $T \vdash \varepsilon(q)$ and $T\varepsilon(r) \vdash \varepsilon(s)$ where all the propositions are F-propositions and all the witnesses F-terms. From the proof π'_1 we build a proof π' of $T \vdash \varepsilon(s)$ under the hypothesis $T \vdash \varepsilon(r)$ where all the propositions are F-propositions and all the witnesses F-terms. We build the proof

$$\dfrac{\dfrac{\dfrac{}{T \vdash \varepsilon(p)}\text{axiom} \quad \dfrac{\rho'}{T \vdash \varepsilon(q)}}{T \vdash \varepsilon(r)}}{\dfrac{\pi'}{T \vdash \varepsilon(s)}}$$

  If this rule in an elimination of the universal quantifier then the proof has the form

$$\dfrac{\dfrac{\dfrac{}{T \vdash \varepsilon(p)}\text{axiom}}{T \vdash Q}(x, R, t)\ \forall\text{-elim}}{\dfrac{\pi}{T \vdash \varepsilon(s)}}$$

  where $\varepsilon(p) \equiv \forall x\ R$ and $Q \equiv \{t/x\}R$. Hence $p \equiv (\dot{\forall}\ r)$, $R \equiv \varepsilon(r\ x)$, and $Q \equiv \varepsilon(r\ t)$. Call σ the substitution mapping each variable x of t of sort $A_1, \ldots, A_n \vdash B$ to the term $x'[\uparrow^n]$ where x' is a fresh variable of sort $\vdash B$. By induction on the structure of π, the proof $\sigma\pi$ is a proof of $T \vdash \varepsilon(s)$ under the hypothesis $T \vdash \sigma Q$. From this proof, we can build a proof π_1 of $T\varepsilon(r\ \sigma t) \vdash \varepsilon(s)$ by adding the proposition $\varepsilon(r\ \sigma t)$ to all the contexts and adding an axiom rule on the top. We apply the induction hypothesis to this proof. Hence there is a proof π'_1 of $T\varepsilon(r\ \sigma t) \vdash \varepsilon(s)$ where all the propositions are F-propositions and all the witnesses F-terms. From this proof we can build a proof π' of $T \vdash \varepsilon(s)$ under the hypothesis $T \vdash \varepsilon(r\ \sigma t)$ where all propositions are F-propositions and all the witnesses F-terms. We build the proof

$$\dfrac{\dfrac{\dfrac{}{T \vdash \varepsilon(p)}\text{axiom}}{T \vdash \varepsilon(r\ \sigma t)}(x, \varepsilon(r\ x), \sigma t)\ \forall\text{-elim}}{\dfrac{\pi'}{T \vdash \varepsilon(s)}}$$

The other cases are similar.

□

Now we can give the proof of our main result.

Proof. The direct sense is an easy induction on the structure of the proof in HOL-λ. An an example, we give the case of the last rule is an elimination of the universal quantifier. The proof has the form

$$\cfrac{\cfrac{\pi}{T \vdash p}}{T \vdash q}\ (r,t)\ \dot{\forall}\text{-elim}$$

where $p \equiv (\dot{\forall}\ r)$ and $q \equiv (r\ t)$. Then $\varepsilon(p_F) = \varepsilon(\dot{\forall}\ r_F) \equiv \forall x\ \varepsilon(r_F\ x)$. By induction hypothesis, there is a proof π' of the proposition $\varepsilon(p_F)$. We build the proof

$$\cfrac{\cfrac{\pi'}{T_F \vdash \varepsilon(p_F)}}{T \vdash \varepsilon(q_F)}\ (x, \varepsilon(r_F\ x), t_F)\ \forall\text{-elim}$$

Conversely, by the proposition 5.4, we can build a proof of $\varepsilon(p_{1F}), \ldots, \varepsilon(p_{nF}) \vdash \varepsilon(q_F)$ where all the propositions are F-propositions and all the witnesses F-terms. By induction on the structure of this proof we can build a proof of $p_1, \ldots, p_n \vdash q$ in HOL-λ. As an example, we give the case of the rule $\forall$-elim. The proof has the form

$$\cfrac{\cfrac{\pi}{T_F \vdash \varepsilon(p_F)}}{T_F \vdash \varepsilon(q_F)}\ (x, \varepsilon(r_F), t_F)\ \forall\text{-elim}$$

where $\varepsilon(p_F) = \forall x\ \varepsilon(r_F)$ and $q_F \equiv \{x \mapsto t_F\}r_F$ Hence $p_F \equiv (\dot{\forall}\ s_F)$ and $r_F \equiv (s_F\ x)$ and $q_F \equiv (s_F\ t_F) = (s\ t)_F$. By induction hypothesis, there exists a proof π' in HOL-λ of $T \vdash P$. We build the proof

$$\cfrac{\cfrac{\pi'}{T \vdash p}}{T \vdash q}\ (s,t)\ \dot{\forall}\text{-elim}$$

□

6 Skolemization in HOL-$\lambda\sigma$

Skolemization in higher-order logic is known to be more complicated than in first-order logic. Indeed, the naive skolemization rule in higher-order logic permits to transform some unprovable formulations of the axiom of choice into provable propositions. Thus the naive skolemization rule has to be restricted in such a way that skolemizing a proposition of the form

$$\forall x_1 \ldots \forall x_n \exists y\ P(x_1, \ldots, x_n, y)$$

introduces a skolem symbol f^n that can only be used to be applied to n terms and moreover the variables free in these terms cannot be bound higher in the term. For instance the term $\lambda y\ (f^1\ x\ y)$ is correct, while the terms f^1, $(F\ f^1)$ and $\lambda x\ (f^1\ x\ y)$ are not (Miller's conditions) [Mil83, Mil87].

A motivation for expressing higher-order logic as a first-order theory is to reuse the usual first-order skolemization rule. When we apply the first-order skolemization rule to the first-order presentation of higher-order logic with combinators we get conditions on Skolem symbols that are related to Miller's conditions, but are different because the translation of λ-calculus is not straightforward. We show below that in HOL-$\lambda\sigma$ the conditions we get are exactly Miller's conditions.

6.1 Miller's theorem

The naïve treatment of skolemization in higher-order logic, that skolemizes

$$\forall x\ \exists y\ (P\ x\ y)$$

as

$$\forall x\ (P\ x\ (f\ x))$$

is where f is a constant of type $T \to U$ (where T is the type of x and U that of y) is unsound. Indeed the axiom of choice

$$\forall x\ \exists y\ (P\ x\ y) \Rightarrow \exists g\ \forall x\ (P\ x\ (g\ x))$$

is not provable in type theory [And72]. Thus from the proposition

$$\forall x\ \exists y\ (P\ x\ y)$$

we cannot deduce

$$\exists g\ \forall x\ (P\ x\ (g\ x))$$

While naively skolemizing it yields

$$\forall x\ (P\ x\ (f\ x))$$

from which we can obviously deduce

$$\exists g\ \forall x\ (P\ x\ (g\ x))$$

Miller [Mil83, Mil87] has proposed an alternative skolemization rule by adding two conditions:

- (necessary arguments) the symbol f can be used only applied to its arguments (e.g. $(f\ x)$ is a term, but f alone is not).
- (no bound variables in necessary arguments) the variables free in the necessary arguments cannot be bound by a λ higher in the term) (e.g. $\lambda x\ (f\ y)$ is a term, but $\lambda x\ (f\ x)$ is not).

Notice however that, if we consider as it is usual in higher-order logic that $\forall x\ P$ is a notation for the term $\dot\forall\ (\lambda x\ P)$ where $\dot\forall$ is a constant, then the skolemized proposition $\forall x\ (P\ x\ (f^1\ x))$ itself does not verify the conditions. Hence, we must either introduce quantifiers as new binders or give a more restricted form to Skolem theorem. If we use skolemization to put a proposition to be refuted in clausal form, then the universal quantifier also will be suppressed yielding the proposition $(P\ X\ (f^1\ X))$ where X is a free variable and we can state Skolem theorem as the correctness of this transformation with respect to resolution (i.e. the clausal form of a proposition can be refuted by resolution if and only if the proposition itself is provable in natural deduction). In [Mil83, Mil87] Miller formulates his theorem as the correctness of this transformation with respect to the connection method.

6.2 Combinators

Skolem theorem applies to the first-order presentation of higher-order logic with combinators as it applies to any first-order theory. A proposition of the form

$$\forall x\ \exists y\ (P\ x\ y)$$

is skolemized as

$$\forall x\ (P\ x\ f(x))$$

but then f is not a constant of type $T \to U$ but a function symbol of rank $(T)U$. Hence f alone is not a term (as $+$ is not a term in first-order arithmetics) but $f(x)$ is. We get this way Miller's first condition. As with combinators there is no bound variables, the second condition vanishes in this presentation.

6.3 HOL-$\lambda\sigma$

Skolem theorem also applies to HOL-$\lambda\sigma$ as it applies to any first-order theory. A proposition of the form

$$\forall x\ \exists y\ (P\ x\ y)$$

is skolemized as

$$\forall x\ (P\ x\ f(x))$$

Again f is a unary function symbol and hence we get back Miller's first condition, but its rank is now $(\Gamma \vdash T)\Delta \vdash U$ which expresses Miller's second condition.

For instance the proposition

$$\forall x\ \exists y\ \varepsilon(P\ x\ y)$$

is skolemized as

$$\forall x\ \varepsilon(P\ x\ f(x))$$

where f has rank $(\vdash T) \vdash U$ and this way the term $\lambda(f(1))$ is not well-typed. Indeed, the argument of f must be well-typed in the empty context and 1 is not.

7 Automated theorem proving in HOL-$\lambda\sigma$

We are now able to wrap-up together the above ingredients to get a first-order presentation of higher-order resolution. To this end, as with any first-order theory modulo, we can use the method developed in [DHK98] to search proofs in HOL-$\lambda\sigma$. This method is complete because HOL-$\lambda\sigma$ enjoys the cut elimination property.

Working modulo a congruence introduces two new features with respect to first-order resolution. First, unification is replaced by equational unification, and second, when the rewrite system contains rules rewriting atomic propositions to non-atomic ones, the rule **Extended Narrowing** presented in figure 5 instantiates appropriately the variables.

In [DHK98], we have applied this method to a first-order expression of higher-order logic using combinators and we have shown that the **Extended Narrowing** rule specializes exactly to the **Splitting** rule of higher-order resolution. Unfortunately equational unification modulo the conversion axioms of combinators is not higher-order unification.

$$\frac{\{A_1,\dots,A_n,B_1,\dots,B_m\}\,[E_1] \quad \{\neg C_1,\dots,\neg C_p,D_1,\dots,D_q\}\,[E_2]}{\{B_1,\dots,B_m,D_1,\dots,D_q\}\,[E_1 \cup E_2 \cup \{A_1 =^?_{\mathcal{E}} A_2 \dots =^?_{\mathcal{E}} A_n =^?_{\mathcal{E}} C_1 \dots =^?_{\mathcal{E}} C_p\}]}\ \textbf{Resolution}$$

$$\frac{C\,[E]}{c\ell(C[r]_p)\,[E \cup \{C_{|p} =^?_{\mathcal{E}} l\}]}\ \textbf{Narrowing} \quad \text{if } l \to r \in \mathcal{R} \text{ and } C_{|p} \notin \mathcal{X}$$

Figure 5: Extended narrowing and resolution (ENAR)

If we apply this method to HOL-$\lambda\sigma$, the **Extended Narrowing** rule still specializes to the **Splitting** rule of higher-order resolution [Hue72, Hue73], but the unification required is the unification modulo the system $\lambda\sigma$ that we have shown to be equivalent to higher-order unification in [DHK95]. Thus, the method obtained this way simulates higher-order resolution step by step.

Conclusion

In this paper we have given a first-order presentation of higher-order logic. This presentation is intentionally equivalent to the presentation of higher-order logic based on λ-calculus. Applying the Extended Narrowing and Resolution method to this theory gives exactly higher-order resolution. Hence we show this way that expressing higher-order logic as a first-order theory and applying a first-order proof search method does not necessarily lead to inefficiencies, provided we take the good first-order expression of higher-order logic and the good proof search method.

Expressing higher-order resolution in a first-order framework permits to clarify its features : higher-order unification, the splitting rule and higher-order resolution. Higher-order resolution is equational unification in an appropriate theory. The splitting rule is an instance of the extended narrowing rule introduced in [DHK98], it is needed because the rewrite system of higher-order logic transforms atomic propositions into non atomic ones. The higher-order skolemization rule is an instance of the first-order one. Its scoping particularities are consequences of the sort system of higher-order logic.

As we stay in a first-order setting, we can also reuse optimizations of first-order theorem proving such as redundancy criterias, subsumption, ...

As we stay in a first-order setting, extending the method to equational higher-order resolution requires only to add more reduction rules to the rewrite system $\lambda\sigma\mathcal{L}$, then narrowing provides an equational higher-order unification algorithm [KR97] and the proof search method is complete provided deduction modulo the extended theory verifies the cut elimination property.

References

[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

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

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

[And86] P.B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof.* Academic Press inc., New York, 1986.

[Chu40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[DHK95] G. Dowek, Th. Hardin, and C. Kirchner. Higher-order unification via explicit substitutions, extended abstract. In D. Kozen, editor, *Proceedings of LICS'95*, pages 366–374, San Diego, June 1995.

[DHK98] G. Dowek, Th. Hardin, and C. Kirchner. Theorem proving modulo. Rapport de Recherche 3400, Institut National de Recherche en Informatique et en Automatique, April 1998. `http://coq.inria.fr/~dowek/RR-3400.ps.gz`.

[DW98] G. Dowek and B. Werner. Proof normalization modulo. Rapport de Recherche 3542, Institut National de Recherche en Informatique et en Automatique, November 1998. `http://coq.inria.fr/~dowek/RR-3542.ps.gz`.

[Gir70] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse et son application à l'élimination des coupures dans l'analyse et la théorie des types. In *J.E. Fenstad (Ed.), Second Scandinavian Logic Symposium.* North-Holland, 1970.

[Gir72] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur.* PhD thesis, Paris VII, 1972.

[GL97] J. Goubault-Larrecq. A proof of weak termination of the simply-typed $\lambda\sigma$-calculus. Technical Report 3090, INRIA, January 1997.

[Hin64] J.R. Hindley. *The Church-Rosser Property and a Result in Combinatory Logic.* PhD thesis, University of Newcastle-upon-Tyne, 1964.

[Hue72] G. Huet. *Constrained Resolution: A Complete Method for Type Theory.* PhD thesis, Case Western Reserve University, 1972.

[Hue73] G. Huet. The undecidability of unification in third order logic. *Information and Control*, 22:257–267, 1973.

[KR97] C. Kirchner and Ch. Ringeissen. Higher-Order Equational Unification via Explicit Substitutions. In *Proceedings 6th International Joint Conference ALP'97-HOA'97, Southampton (UK)*, volume 1298 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 1997.

[KvOvR93] J.W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.

[Mil83] D.A. Miller. *Proofs in higher order logic.* PhD thesis, Carnegie Mellon University, 1983.

[Mil87] D.A. Miller. A compact representation of proofs. *Studia Logica*, XLVI(4):347–370, 1987.

[Muñ97] C. Muñoz. A left linear variant of $\lambda\sigma$. In *Proceedings 6th International Joint Conference ALP'97-HOA'97, Southampton (UK)*, volume 1298 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[Plo72] G. Plotkin. Building-in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[PS81] G. Peterson and M.E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28:233–264, 1981.

Contents

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399