HOL-λσ: an intentional first-order expression of higher-order logic

GILLES DOWEK, THERESE HARDIN and CLAUDE KIRCHNER

Mathematical Structures in Computer Science / Volume 11 / Issue 01 / February 2001, pp 21 - 45
DOI: null, Published online: 07 March 2001

Math. Struct. in Comp. Science (2001), *vol.* 11, *pp.* 21–45. Printed in the United Kingdom

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

GILLES DOWEK[†], THERESE HARDIN[‡] and CLAUDE KIRCHNER[§]

[†] *INRIA-Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France*
Email: Gilles.Dowek@inria.fr, http://coq.inria.fr/~dowek

[‡] *LIP6 & INRIA, UPMC, 4 place Jussieu, 75252 Paris Cedex 05, France*
Email: Therese.Hardin@lip6.fr, http://www-spi.lip6.fr/~hardin

[§] *LORIA & INRIA, 615, rue du Jardin Botanique, 54600 Villers-lès-Nancy, France*
Email: Claude.Kirchner@loria.fr, http://www.loria.fr/~ckirchne

Received 18 October 1999; revised 3 April 2000

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, that is, a proposition can be proved without the extensionality axioms in one theory if and only if it can be in the other. We show that the *Extended Narrowing and Resolution* first-order proof-search method can be applied to this theory. In this way we get 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 a relevant alternative to a direct implementation. In particular, 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, may be easier to handle.

1. 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 the automation of proof search in this logic.

Higher-order logic can be expressed in many different ways using combinators, λ-calculus, *etc*. Some of these formulations, but not all, present higher-order logic as a first-order theory. Such a formulation allows us to use standard first-order methods for proof search and it may also allow us to handle extensions more easily. 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. For instance, higher-order logic can be encoded as a first-order theory using combinators. The theory obtained in this way is equivalent to the standard presentation using λ-calculus, but it is not *intentionally*

equivalent to it: some proofs in the theory use the extensionality axioms with combinators, but do not in the standard presentation. Moreover, the additional use of the extensionality axioms, which express the fact that two pointwise equivalent functions are equal and that two sets that have the same elements are equal, induces inefficiencies in proof search.

In this paper, which is a revised and extended version of Dowek *et al.* (1999) and builds upon Dowek *et al.* (2000) and Dowek *et al.* (1998), we give a new first-order presentation of higher-order logic called HOL-$\lambda\sigma$. It uses the fact, which has already been noted by several authors, that explicit substitutions simplify algorithms and speed up implementations (Nadathur and Wilson 1990; Magnusson 1994; Dowek *et al.* 1996; Dowek *et al.* 2000; Muñoz 1997b; Nadathur and Wilson 1998). Using the *calculus of explicit substitutions*, which has been defined and studied in Abadi *et al.* (1991) and Curien *et al.* (1996), we first define HOL-$\lambda\sigma$ and show that it is intentionally equivalent to the usual presentation of higher-order logic based on λ-calculus, that is, the theories are still equivalent when we drop the extensionality axioms in both cases.

We then show that proof-search in this theory can be mechanized with the *Extended Narrowing and Resolution* (ENAR) method introduced in Dowek *et al.* (1998). The proof search method for higher-order logic obtained in this way is as efficient as higher-order resolution and in fact simulates it step by step: proof search steps correspond and β-reduction steps correspond to $\lambda\sigma$-reduction steps. It maintains, however, the simplicity of first-order frameworks and could easily be extended, for instance with equational axioms.

Finally, 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 (Miller 1983; Miller 1987).

HOL-$\lambda\sigma$ and the ENAR proof search method rely on a presentation of first-order logic called *deduction modulo* that allows one to build in a congruence identifying not only terms but also propositions. This leads to shorter and more direct proofs by identifying congruent propositions instead of requiring explicit equivalence proofs. Hence, we shall express HOL-$\lambda\sigma$ in deduction modulo. In order to make the paper self-contained, we recall the principal ideas of deduction modulo in Section 2. Then, we recall in Section 3 the usual presentation of higher-order logic based on λ-calculus (HOL-λ) and in Section 4 its first-order presentation based on combinators. Section 5 introduces HOL-$\lambda\sigma$ and establishes its main properties (termination, confluence, consistency and cut elimination). Section 6 is dedicated to the equivalence theorem between HOL-λ and HOL-$\lambda\sigma$ (which rests upon cut elimination). In Section 7 we show that the rather intricate Skolem theorem for higher-order logic can be deduced from the first-order one. Finally, Section 8 presents the ENAR proof search method (whose completeness rests also upon cut elimination) and its application to HOL-$\lambda\sigma$.

2. Deduction modulo

In this paper we shall use a presentation of first-order logic, called *deduction modulo* (Dowek *et al.* 1998), which allows us 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* formed with a set of axioms Γ *and a*

congruence, denoted $\equiv$, defined on propositions. The deduction rules take this equivalence into account, for instance, the right rule of the conjunction is not given by the usual

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

but is formulated as

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash C, \Delta} \text{ if } C \equiv A \wedge B$$

and all the rules of sequent calculus are stated in a similar way, as described in Figure 1.

When Γ is a finite set of axioms and $\equiv$ a congruence, a proposition P is said to be provable in $(\Gamma, \equiv)$ if the sequent $\Gamma \vdash P$ is derivable modulo $\equiv$. When Γ is infinite, a proposition P is said to be provable in $(\Gamma, \equiv)$ if it is provable in $(\Gamma', \equiv)$, where Γ' is a finite subset of Γ.

For instance, in sequent calculus modulo the congruence defined by the rewrite system

$$\begin{aligned} 0 + y &\longrightarrow y \\ S(x) + y &\longrightarrow S(x + y) \\ 0 \times y &\longrightarrow 0 \\ S(x) \times y &\longrightarrow x \times y + y \end{aligned}$$

we can prove that the number 4 is even.

$$\dfrac{\dfrac{\dfrac{}{4 = 4 \vdash 2 \times 2 = 4}\,\text{axiom}}{\forall x\ x = x \vdash 2 \times 2 = 4}\,(x, x = x, 4)\ \forall\text{-l}}{\forall x\ x = x \vdash \exists y\ 2 \times y = 4}\,(y, 2 \times y = 4, 2)\ \exists\text{-r}$$

Substituting the variable y by the term 2 in the proposition $2 \times y = 4$, as indicated on the side of the rule, yields the proposition $2 \times 2 = 4$, that is congruent to $4 = 4$. The transformation of one proposition into the other, which would require several proof steps in sequent calculus using the axioms of arithmetic, does not appear here. Since the congruence is decidable, this computation does not need to be recorded in the proof itself.

Notice that we do not use the axioms of addition and multiplication explicitly in the proof. Indeed, these axioms are now redundant: as 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. Using the terminology introduced by Plotkin, these axioms have been *built in* (Plotkin 1972; Andrews 1971; Peterson and Stickel 1981; Stickel 1985; Jouannaud and Kirchner 1986; Marché 1994; Viry 1995; Viry 1998).

In the example above, the congruence is just the congruent closure of the relation induced on terms by the term rewriting system. In many situations, it is also natural to consider congruences defined directly at the proposition level. For instance, we may add to the previous system the rule of integral domains

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

that rewrites an atomic proposition to a disjunction. As far as we know, this property cannot be expressed by term rewriting rules. Deduction modulo can consider congruences

$$\frac{}{P \vdash Q}\text{axiom} \quad \text{if } P \equiv Q \qquad \frac{\Gamma, P \vdash \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash \Delta}\text{cut} \quad \text{if } P \equiv Q$$

$$\frac{\Gamma, Q_1, Q_2 \vdash \Delta}{\Gamma, P \vdash \Delta}\text{contr-l} \quad \text{if } P \equiv Q_1 \equiv Q_2 \qquad \frac{\Gamma \vdash Q_1, Q_2, \Delta}{\Gamma \vdash P, \Delta}\text{contr-r} \quad \text{if } P \equiv Q_1 \equiv Q_2$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, P \vdash \Delta}\text{weak-l} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash P, \Delta}\text{weak-r}$$

$$\frac{\Gamma \vdash P, \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, R \vdash \Delta}\Rightarrow\text{-l} \quad \text{if } R \equiv (P \Rightarrow Q) \qquad \frac{P, \Gamma \vdash Q, \Delta}{\Gamma \vdash R, \Delta}\Rightarrow\text{-r} \quad \text{if } R \equiv (P \Rightarrow Q)$$

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, R \vdash \Delta}\wedge\text{-l} \quad \text{if } R \equiv (P \wedge Q) \qquad \frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash R, \Delta}\wedge\text{-r} \quad \text{if } R \equiv (P \wedge Q)$$

$$\frac{\Gamma, P \vdash \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, R \vdash \Delta}\vee\text{-l} \quad \text{if } R \equiv (P \vee Q) \qquad \frac{\Gamma \vdash P, Q, \Delta}{\Gamma \vdash R, \Delta}\vee\text{-r} \quad \text{if } R \equiv (P \vee Q)$$

$$\frac{\Gamma \vdash P, \Delta}{\Gamma, R \vdash \Delta}\neg\text{-l} \quad \text{if } R \equiv \neg P \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash R, \Delta}\neg\text{-r} \quad \text{if } R \equiv \neg P$$

$$\frac{}{\Gamma, P \vdash \Delta}\bot\text{-l} \quad \text{if } P \equiv \bot$$

$$\frac{\Gamma, \{t/x\}P \vdash \Delta}{\Gamma, Q \vdash \Delta}(x, P, t)\ \forall\text{-l} \quad \text{if } Q \equiv \forall x\ P \qquad \frac{\Gamma \vdash \{y/x\}P, \Delta}{\Gamma \vdash Q, \Delta}(x, P, y)\ \forall\text{-r} \quad \text{if } Q \equiv \forall x\ P$$

$$\frac{\Gamma, \{y/x\}P \vdash \Delta}{\Gamma, Q \vdash \Delta}(x, P, y)\ \exists\text{-l} \quad \text{if } Q \equiv \exists x\ P \qquad \frac{\Gamma \vdash \{t/x\}P, \Delta}{\Gamma \vdash Q, \Delta}(x, P, t)\ \exists\text{-r} \quad \text{if } Q \equiv \exists x\ P$$

where the rules $\forall$-r and $\exists$-l assume that $y \notin FV(\Gamma\Delta)$

Fig. 1. The sequent calculus modulo

on propositions defined by rules rewriting terms to terms and atomic propositions to arbitrary ones. Rules with non-atomic left-hand sides, which are technically much more difficult to handle, are not tackled in this work.

All congruences in this paper are 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* (Klop *et al.* 1993).

Notice that deduction modulo is not a proper extension of first-order logic. It is proved in Dowek *et al.* (1998) that for every congruence $\equiv$, we can find a theory $\mathscr{T}$ such that $\Gamma \vdash P$ is provable modulo $\equiv$ if and only if $\mathscr{T}, \Gamma \vdash P$ is provable in ordinary first-order logic. Of course, the provable propositions are the same, but the proofs are very different, indeed much shorter in deduction modulo.

Proof search in deduction modulo can be handled by a method called *Extended Narrowing and Resolution* (ENAR), which extends the usual resolution method and is described below.

3. HOL-λ

In this section we run quickly through the usual presentation of higher-order logic. Terms are those of a simply typed λ-calculus (Girard *et al.* 1989) with two base types ι and o and the following constants:

— $\dot{\Rightarrow}$, $\dot{\wedge}$ and $\dot{\vee}$ of type $o \to o \to o$,
— $\dot{\neg}$ of type $o \to o$,
— $\dot{\bot}$ of type o,
— for each type T, constants $\dot{\forall}_T$ and $\dot{\exists}_T$ of type $(T \to o) \to o$.

The notation with a dot for the constants lets us distinguish them from the connectors and quantifiers of first-order logic. Terms of type o are called *propositions*.

The unique $\beta\eta$-normal form of a term a is written $a\downarrow$. The deduction rules of HOL-λ are given in Figure 2 where all propositions are supposed to be $\beta\eta$-normal.

For instance, we can prove the sequent

$$(\dot{\forall}\ \lambda P\ (\dot{\Rightarrow}\ (P\ a)\ (P\ b))), (R\ a\ a) \vdash (R\ b\ b)$$

as follows:

$$\dfrac{\dfrac{\dfrac{}{(R\ a\ a) \vdash (R\ a\ a)}\ \text{axiom} \qquad \dfrac{}{(R\ b\ b) \vdash (R\ b\ b)}\ \text{axiom}}{(\dot{\Rightarrow}\ (R\ a\ a)\ (R\ b\ b)), (R\ a\ a) \vdash (R\ b\ b)}\ \dot{\Rightarrow}\text{-l}}{(\dot{\forall}\ \lambda P\ (\dot{\Rightarrow}\ (P\ a)\ (P\ b))), (R\ a\ a) \vdash (R\ b\ b)}\ \dot{\forall}\text{-l}$$

where in the $\dot{\forall}$-left rule, the term $\lambda P\ (\dot{\Rightarrow}\ (P\ a)\ (P\ b))$ has been applied to the term $\lambda x\ (R\ x\ x)$.

In an alternative presentation of HOL-λ, propositions are not normalized in the quantifier rules. Instead, such a presentation takes axioms stating that two $\beta\eta$-convertible terms are equal, see, for instance, Church (1940) and Andrews (1986). In this case, for the example above, applying the term $\lambda P\ (\dot{\Rightarrow}\ (P\ a)\ (P\ b))$ to the term $\lambda x\ (R\ x\ x)$, we get $(\lambda P\ (\dot{\Rightarrow}\ (P\ a)\ (P\ b))\ \lambda x\ (R\ x\ x))$, and we use the axioms to deduce $(\dot{\Rightarrow}\ (R\ a\ a)\ (R\ b\ b))$.

The system HOL-λ is well-known to be consistent and to enjoy cut elimination (Girard 1970; Girard 1972).

In HOL-λ, equality does not need to be primitive. It can be defined as Leibniz' equality, that is, $\lambda x\ \lambda y\ \dot{\forall}\lambda p\ ((p\ x)\dot{\Rightarrow}(p\ y))$. In this case, the propositions

$$\dot{\forall}\ \lambda f\ \dot{\forall}\ \lambda g\ \dot{\forall}\ \lambda x\ (f = g\ \dot{\Rightarrow}\ (f\ x) = (g\ x))$$
$$\dot{\forall}\ \lambda x\ \dot{\forall}\ \lambda y\ \dot{\forall}\ \lambda f\ (x = y\ \dot{\Rightarrow}\ (f\ x) = (f\ y))$$

are provable. But the proposition

$$\dot{\forall}\ \lambda f\ \dot{\forall}\ \lambda g\ (\dot{\forall}\ \lambda x\ ((f\ x) = (g\ x))\ \dot{\Rightarrow}\ \lambda x\ (f\ x) = \lambda x\ (g\ x))$$

is not, because substitutions avoid captures. Similarly, the propositions

$$\dot{\forall}\ \lambda f\ \dot{\forall}\ \lambda g\ ((\dot{\forall}\ \lambda x\ ((f\ x) = (g\ x)))\ \dot{\Rightarrow}\ f = g)$$

and

$$\dot{\forall}\ \lambda x\ \dot{\forall}\ \lambda y\ ((x \dot{\Leftrightarrow} y)\ \dot{\Rightarrow}\ x = y)$$

are not provable.

$$\frac{}{P \vdash P}\text{axiom} \qquad \frac{\Gamma, P \vdash \Delta \quad \Gamma \vdash P, \Delta}{\Gamma \vdash \Delta}\text{cut}$$

$$\frac{\Gamma, P, P \vdash \Delta}{\Gamma, P \vdash \Delta}\text{contr-l} \qquad \frac{\Gamma \vdash P, P, \Delta}{\Gamma \vdash P, \Delta}\text{contr-r}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, P \vdash \Delta}\text{weak-l} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash P, \Delta}\text{weak-r}$$

$$\frac{\Gamma \vdash P, \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, (\dot{\Rightarrow}\ P\ Q) \vdash \Delta}\dot{\Rightarrow}\text{-l} \qquad \frac{P, \Gamma \vdash Q, \Delta}{\Gamma \vdash (\dot{\Rightarrow}\ P\ Q), \Delta}\dot{\Rightarrow}\text{-r}$$

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, (\dot{\wedge}\ P\ Q) \vdash \Delta}\dot{\wedge}\text{-l} \qquad \frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash (\dot{\wedge}\ P\ Q), \Delta}\dot{\wedge}\text{-r}$$

$$\frac{\Gamma, P \vdash \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, (\dot{\vee}\ P\ Q) \vdash \Delta}\dot{\vee}\text{-l} \qquad \frac{\Gamma \vdash P, Q, \Delta}{\Gamma \vdash (\dot{\vee}\ P\ Q), \Delta}\dot{\vee}\text{-r}$$

$$\frac{\Gamma \vdash P, \Delta}{\Gamma, (\dot{\neg}\ P) \vdash \Delta}\dot{\neg}\text{-l} \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash (\dot{\neg}\ P), \Delta}\dot{\neg}\text{-r}$$

$$\frac{}{\Gamma, \dot{\bot} \vdash \Delta}\dot{\bot}\text{-l}$$

$$\frac{\Gamma, (P\ t)\downarrow \vdash \Delta}{\Gamma, (\dot{\forall}\ P) \vdash \Delta}t\ \dot{\forall}\text{-l} \qquad \frac{\Gamma \vdash (P\ y)\downarrow, \Delta}{\Gamma \vdash (\dot{\forall}\ P), \Delta}\dot{\forall}\text{-r}$$

$$\frac{\Gamma, (P\ y)\downarrow \vdash \Delta}{\Gamma, (\dot{\exists}\ P) \vdash \Delta}\dot{\exists}\text{-l} \qquad \frac{\Gamma \vdash (P\ t)\downarrow, \Delta}{\Gamma \vdash (\dot{\exists}\ P), \Delta}t\ \dot{\exists}\text{-r}$$

where the rules $\dot{\forall}$-r and $\dot{\exists}$-l assume that $y \notin FV(\Gamma\Delta)$

Fig. 2. **HOL-λ**: The deduction rules of HOL-λ

The last two propositions can be added as axioms in the theory. They are called *extensionality axioms* and the theory itself is called *extensional higher-order logic*. In contrast, without these extensionality axioms, the theory is called *intentional higher-order logic*.

4. HOL-C

4.1. *HOL-C as a first-order theory*

Higher-order logic can be expressed as a many-sorted first-order theory with equality. The sorts of this theory are the types of simply typed λ-calculus, that is, they are inductively defined by:

— ι and o are sorts,
— if T and U are sorts, then $T \to U$ is a sort.

A function symbol f is said to have *rank* $(T_1, \ldots, T_n)\ U$ if it takes as arguments n terms of sorts $T_1, \ldots, T_n$ and constructs a term of sort U. A predicate symbol P is said to have *rank* $(T_1, \ldots, T_n)$ if it takes as arguments n terms of sorts $T_1, \ldots, T_n$.

Besides equality, the language contains the function symbols:

— $\alpha_{T,U}$ of rank $(T \to U, T)\ U$.

These symbols are called *applications*. As usual, the term $\alpha_{T,U}(t, u)$ is written $(t\ u)$ and $(\ldots(t\ u_1)\ldots u_n)$ is written $(t\ u_1\ \ldots\ u_n)$.

Then, the language contains the unary predicate symbol:

— ε of rank (o),

which transforms a term t of sort o into the proposition $\varepsilon(t)$.

To express function and predicate terms, instead of using λ-calculus, we introduce for each n-tuple of variables $x_1, \ldots, x_n$, respectively of sorts $T_1, \ldots, T_n$, and each term t of sort U formed with the variables $x_1, \ldots, x_n$ and the application symbols, a constant symbol:

— $x_1, \ldots, x_n \longmapsto t$ of sort $T_1 \to \ldots \to T_n \to U$.

Such constant symbols are called *combinators*. Finally, the language also contains the constant symbols:

— $\dot{\Rightarrow}$, $\dot{\wedge}$ and $\dot{\vee}$ of sort $o \to o \to o$,
— $\dot{\neg}$ of sort $o \to o$,
— $\dot{\bot}$ of sort o,
— $\dot{\forall}_T$ and $\dot{\exists}_T$ of sort $(T \to o) \to o$.

Besides the well-known axioms of equality, the theory contains the following axioms that express the meaning of the combinators:

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

and axioms that relate the connectors and quantifiers (for example, $\wedge$) and their replication as constant symbols (for example, $\dot{\wedge}$):

$$\begin{aligned}
\varepsilon(\dot{\Rightarrow}\ x\ y) &\Leftrightarrow (\varepsilon(x) \Rightarrow \varepsilon(y)) \\
\varepsilon(\dot{\wedge}\ x\ y) &\Leftrightarrow (\varepsilon(x) \wedge \varepsilon(y)) \\
\varepsilon(\dot{\vee}\ x\ y) &\Leftrightarrow (\varepsilon(x) \vee \varepsilon(y)) \\
\varepsilon(\dot{\neg}\ x) &\Leftrightarrow \neg\varepsilon(x) \\
\varepsilon(\dot{\bot}) &\Leftrightarrow \bot \\
\varepsilon(\dot{\forall}\ x) &\Leftrightarrow \forall y\ \varepsilon(x\ y) \\
\varepsilon(\dot{\exists}\ x) &\Leftrightarrow \exists y\ \varepsilon(x\ y).
\end{aligned}$$

Notice that, with our convention for application notation, the term $(\dot{\Rightarrow}\ x\ y)$ is indeed $\alpha(\alpha(\dot{\Rightarrow}, x), y)$ and that $\varepsilon(\dot{\Rightarrow}\ x\ y)$ is an atomic proposition.

4.2. *HOL-C as a first-order theory modulo*

In deduction modulo, these axioms can be built in. So, we work modulo the congruence defined by the rewriting system $\mathscr{R}$ containing the following term rewrite rules:

$$((x_1, \ldots, x_n \longmapsto t)\ u_1\ \ldots\ u_n) \longrightarrow \{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}t$$

and the following proposition rewrite rules:

$$\begin{aligned}
\varepsilon(\dot{\Rightarrow}\ x\ y) &\longrightarrow \varepsilon(x) \Rightarrow \varepsilon(y)\\
\varepsilon(\dot{\wedge}\ x\ y) &\longrightarrow \varepsilon(x) \wedge \varepsilon(y)\\
\varepsilon(\dot{\vee}\ x\ y) &\longrightarrow \varepsilon(x) \vee \varepsilon(y)\\
\varepsilon(\dot{\neg}\ x) &\longrightarrow \neg\varepsilon(x)\\
\varepsilon(\dot{\bot}) &\longrightarrow \bot\\
\varepsilon(\dot{\forall}\ x) &\longrightarrow \forall y\ \varepsilon(x\ y)\\
\varepsilon(\dot{\exists}\ x) &\longrightarrow \exists y\ \varepsilon(x\ y).
\end{aligned}$$

Notice that the rules of the first group rewrite terms, while the rules of the second group rewrite atomic propositions. Hence, we have here a typical example where rules rewriting terms are not enough.

Notice also that in this formulation, equality can be defined as Leibniz' equality. Indeed, if a and b are terms of type T, the notation $a = b$ may be introduced as an abbreviation for the proposition $\forall p\ (\varepsilon(p\ a) \Rightarrow \varepsilon(p\ b))$, where p is a variable of type $T \to o$. Thus there is no need to take equality as a primitive symbol.

4.3. *The extensionality axioms*

In HOL-C, the extensionality axioms are

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

$$\forall x\ \forall y\ (\varepsilon(x) \Leftrightarrow \varepsilon(y)) \Rightarrow x = y)$$

Here equality is either a primitive symbol as in Section 4.1 or an abbreviation as in Section 4.2.

4.4. *Embedding HOL-λ into HOL-C*

The translation from λ-terms to combinators is usually called λ-*lifting* and is denoted by $(\_)_C$. Applications, variables and constants are translated in an obvious way to their correspondents in HOL-C. A term of the form $\lambda x\ t$ is translated as follows. Let $y_1, \ldots, y_n$ be the variables of t_C minus x. Then, in t_C, all the occurrences of any combinator c_i are replaced by a fresh variable z_i yielding a term $(t_C)'$. Then,

$$(\lambda x\ t)_C = ((y_1, \ldots, y_n, z_1, \ldots, z_p, x \longmapsto (t_C)')\ y_1\ \ldots\ y_n\ c_1\ \ldots\ c_p).$$

For instance the term

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

is translated as

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

This translation can be modified in order to use just the combinators $S = x, y, z \longmapsto ((x\ z)\ (y\ z))$ and $K = x, y \longmapsto x$.

Extensional HOL-C can be shown to be equivalent to extensional HOL-λ, that is, a

proposition P is provable in extensional HOL-λ if and only if the proposition $\varepsilon(P_C)$ is provable in extensional HOL-C.

But, if we drop the extensionality axioms, the two presentations are no longer equivalent. For instance, the proposition

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

is provable in HOL-λ, while its translation in HOL-C

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

requires extensionality.

Even for extensional higher-order logic, the formulations with λ-calculus and combinators are only weakly equivalent: provable propositions are the same, but the proofs are very different, since some proofs just requiring $\beta\eta$-conversion in HOL-λ require the use of extensionality in HOL-C.

5. HOL-$\lambda\sigma$

We now present the definition and first properties of the first-order theory modulo HOL-$\lambda\sigma$, and we will see in the next section that it provides a new first-order formulation of higher-order logic. The theory HOL-$\lambda\sigma$ is not based on combinators, as previously, but on de Bruijn indices and explicit substitutions. It allows us to avoid the drawbacks of the formulation with combinators mentioned just above.

5.1. *HOL-λσ as a first-order theory modulo*

In λ-calculus with de Bruijn indices, bound variables are replaced by an index indicating the binding height of this variable, that is, the number of λ's between this occurrence and its binder. For instance the term $\lambda x\ (x\ (\lambda y\ x))$ is written $\lambda\ (1\ (\lambda\ 2))$. So λ-calculus with de Bruijn notation is also a first-order language with a binary function symbol α, a unary function symbol λ and constant symbols $1, 2, 3 \ldots$.

Types of simply typed λ-calculus are no longer sufficient with de Bruijn indices. Indeed, we need to give a sort not only to terms like $(\lambda_T\ 1)$ (which gets the sort $T \to T$), but also to terms of the form 1. Thus, as described in Dowek *et al.* (1995), we have to consider sorts of the form $\Gamma \vdash T$ where T is a type of simply typed λ-calculus and Γ is a context, that is, a list of such types. For example, the term 1 has the sort $A.\Gamma \vdash A$.

With de Bruijn indices only, 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* (Abadi *et al.* 1991), which is also called $\lambda\sigma$-calculus. As well as the sorts of the form $\Gamma \vdash T$, this calculus also introduces sorts of the form $\Gamma \vdash \Delta$ for substitutions that are lists of terms. The symbols to build such substitutions are *id*, ., $\uparrow$ and $\circ$. Then, a new term constructor is introduced $\_[\_]$, which allows us to apply an explicit substitution to a term. The rewrite rules describing the evaluation of the $\lambda\sigma$-calculus are given in

β-reduction and η-reduction:

$$\begin{aligned}(\lambda a)b &\longrightarrow a[b.id]\\ \lambda(a\ 1) &\longrightarrow b \text{ if } a =_\sigma b[\uparrow]\end{aligned}$$

σ-reduction:

$$\begin{aligned}(a\ b)[s] &\longrightarrow (a[s]\ b[s])\\ \mathtt{1}[a.s] &\longrightarrow a\\ a[id] &\longrightarrow a\\ (\lambda a)[s] &\longrightarrow \lambda(a[\mathtt{1}.(s\circ\uparrow)])\\ (a[s])[t] &\longrightarrow a[s\circ t]\\ id\circ s &\longrightarrow s\\ \uparrow\circ(a.s) &\longrightarrow s\\ (s_1\circ s_2)\circ s_3 &\longrightarrow s_1\circ(s_2\circ s_3)\\ (a.s)\circ t &\longrightarrow a[t].(s\circ t)\\ s\circ id &\longrightarrow s\\ \mathtt{1}.\uparrow &\longrightarrow id\\ \mathtt{1}[s].(\uparrow\circ s) &\longrightarrow s\end{aligned}$$

Fig. 3. The rewrite rules of $\lambda\sigma$-calculus

Figure 3. They will be used to define (a part of) the congruence, so they give an example where the congruence is defined from a conditional rewrite system.

Now we introduce HOL-$\lambda\sigma$. It is a many-sorted first-order theory modulo with sorts of the form $\Gamma \vdash T$ and $\Gamma \vdash \Delta$ where Γ and Δ are sequences of types of simply typed λ-calculus and T is such a type.

Definition 5.1. (Language of HOL-$\lambda\sigma$) The language contains the following function symbols:

$\mathtt{1}_A^\Gamma$	constant of sort	$A.\Gamma \vdash A$
$\alpha_{A\to B,A}^\Gamma$	binary function of rank	$(\Gamma \vdash A\to B, \Gamma\vdash A)\Gamma\vdash B$
$\lambda_{A,B}^\Gamma$	unary function of rank	$(A.\Gamma\vdash B)\Gamma\vdash A\to B$
$[\ ]_A^{\Gamma,\Gamma'}$	binary function of rank	$(\Gamma'\vdash A, \Gamma\vdash\Gamma')\Gamma\vdash A$
id^Γ	constant of sort	$\Gamma\vdash\Gamma$
$\uparrow_A^\Gamma$	constant of sort	$A.\Gamma\vdash\Gamma$
$\cdot_A^{\Gamma,\Gamma'}$	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	$\vdash (A\to o)\to o$

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

Fig. 4. The $\mathscr{L}$-rewrite rules

and a single unary predicate symbol:

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

We use $\lambda\sigma\mathscr{L}$ to denote the rewrite rules of $\lambda\sigma$-calculus together with the logical rules $\mathscr{L}$ given in Figure 4, and we write $A \equiv_{\lambda\sigma\mathscr{L}} B$ when A and B are congruent modulo $\lambda\sigma\mathscr{L}$.

5.2. *HOL-λσ as a first-order theory*

We have presented HOL-$\lambda\sigma$ as a first-order theory modulo, and it will be used later in that way to represent HOL-λ. We have proved in Dowek *et al.* (1998) that any theory modulo $(\Gamma, \equiv)$ can be also expressed as a (non-modulo) theory, that is, that there exists a set of axioms $\mathscr{T}$ such that $\Gamma \vdash P$ modulo $\equiv$ if and only if $\mathscr{T}, \Gamma \vdash P$ in standard first-order logic.

As with any theory modulo, HOL-$\lambda\sigma$ can be expressed as a first-order theory. The naive expression takes as axioms the universal closures of all propositions of the form $P \Leftrightarrow Q$ where $P \equiv_{\lambda\sigma\mathscr{L}} Q$.

In Dowek *et al.* (1998) we have also shown that when the congruence is defined by rewrite rules, we can take fewer axioms: we first take an equality predicate and the axioms of equality, then for each rewrite rule $l \to r$, we take as axiom the universal closure of the proposition $l = r$ when l and r are terms or $l \Leftrightarrow r$ when l and r are propositions. This result does not apply here because the rule η is a conditionnal rewrite rule. However, for this rule we can take the axiom

$$\forall x\ (\lambda(x[\uparrow]\ 1) = x),$$

and we get a presentation of HOL-$\lambda\sigma$ as a (non modulo) first-order theory.

5.3. *Properties of HOL-λσ*

Proposition 5.1. (Termination) The system $\lambda\sigma\mathscr{L}$ is weakly terminating.

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

Typed $\lambda\sigma$-calculus may not terminate (Melliès 1995), but it is known that the strategy σ-normalizing the term after each application of β or η (Goubault-Larrecq 1997; Muñoz 1997a) is normalizing. We reduce termination for $\lambda\sigma\mathscr{L}$ to termination for this normalizing strategy of $\lambda\sigma$. So, we define a translation of the terms and the propositions of HOL-$\lambda\sigma$

into the typed system $\lambda\sigma$, denoted by $||\_||$, as follows. In each sort $\Gamma \vdash T$, we choose a variable $z_{\Gamma\vdash T}$.

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

In $\lambda\sigma\mathscr{L}$, we say that $t\ R_1$-reduces to u if u is obtained by reducing a β-redex, an η-redex or a $\mathscr{L}$-redex and σ-normalizing the term obtained in this way. In $\lambda\sigma$, we say that $t\ R_2$-reduces to u if u is obtained by reducing a β-redex or an η-redex and σ-normalizing the term obtained in this way. 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 above system, the sequence $||P_1||, ||P_2||, \ldots$ is a R_2-reduction sequence in $\lambda\sigma$, thus it is finite. □

Proposition 5.2. (Confluence) $\lambda\sigma\mathscr{L}$ is confluent on terms containing only term variables.

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

The proof is based on the Hindley–Rosen Lemma (Hindley 1964; Rosen 1973): if two relations R and S are strongly confluent (that is, if $t\ R\ u$ and $t\ R\ v$, there exists a w such that $u\ R\ w$ and $v\ R\ w$, and if $t\ S\ u$ and $t\ S\ v$, there exists a w such that $u\ S\ w$ and $v\ S\ w$) and strongly commute (that is, if $t\ R\ u$ and $t\ S\ v$, there exists a term w such that $u\ S\ w$ and $v\ R\ w$), then the relation $R \cup S$ is confluent.

For R we take $\mathscr{L}$, and for S we take $\lambda\sigma^*$. The system $\mathscr{L}$ is linear and orthogonal, hence it is strongly confluent. Since $\lambda\sigma$ is confluent (Abadi *et al.* 1991) the rewrite relation $\lambda\sigma^*$ is strongly confluent. Finally, $\mathscr{L}$ and $\lambda\sigma^*$ strongly commute. Indeed, if $(t\ \mathscr{L}\ u)$ and $(t\ \lambda\sigma^*\ v)$ then the $\mathscr{L}$-redex in t is either disjoint from or above the $\lambda\sigma$ redex. In both cases we can reduce the $\lambda\sigma$-redex in u and the $\mathscr{L}$ redex in v getting the same term. Hence, $\lambda\sigma\mathscr{L}$ is confluent. □

Proposition 5.3. (Consistency) The theory HOL-$\lambda\sigma$ is consistent.

Proof. We construct a model as follows:

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

If f is a function of the set $(\mathscr{M}_{U_1} \times \ldots \times \mathscr{M}_{U_p})^{\mathscr{M}_{T_n} \ldots \mathscr{M}_{T_1}}$ and g a function of the set $(\mathscr{M}_{V_1} \times \ldots \times \mathscr{M}_{V_q})^{\mathscr{M}_{U_p} \ldots \mathscr{M}_{U_1}}$, we write $g \circ f$ for the function of $(\mathscr{M}_{V_1} \times \ldots \times \mathscr{M}_{V_q})^{\mathscr{M}_{T_n} \ldots \mathscr{M}_{T_1}}$ 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:

— $\overline{1_A^\Gamma}$ is the function mapping $a_1, \ldots, a_n$ to a_1.

— $\overline{\alpha_{A \to B, A}^\Gamma}$ is the function mapping a and b to the function mapping $c_1, \ldots, c_n$ to $a(c_1, \ldots, c_n)(b(c_1, \ldots, c_n))$.

— $\overline{\lambda_{A,B}^\Gamma}$ 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 $a_1, \ldots, a_n$ to $(a_2, \ldots, a_n)$.

— $\overline{\cdot_A^{\Gamma,\Gamma'}}$ is the function mapping a, b to the function mapping $c_1, \ldots, c_n$ to $(a(c_1, \ldots, c_n), b(c_1, \ldots, c_n))$.

— $\overline{\circ^{\Gamma,\Gamma',\Gamma''}}$ 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.

— $\overline{\dot{\wedge}}$ maps a and b to 1 if $a = 1$ and $b = 1$ and to 0 otherwise.

— $\overline{\dot{\vee}}$ maps a and b to 1 if $a = 1$ or $b = 1$ and to 0 otherwise.

— $\overline{\dot{\neg}}$ maps a to 1 if $a = 0$ and to 0 otherwise.

— $\overline{\dot{\bot}} = 0$.

— $\overline{\dot{\forall}_T}$ maps a to 1 if a maps every object of $\mathscr{M}_T$ to 1 and to 0 otherwise.

— $\overline{\dot{\exists}_T}$ maps a to 1 if a maps some object of $\mathscr{M}_T$ to 1 and to 0 otherwise.

— $\bar{\varepsilon}$ is the identity function.

We check that if $A \equiv_{\lambda\sigma\mathscr{L}} B$, then A and B have the same denotation. Then we check that every provable proposition denotes the truth value 1 and hence that $\bot$ is not derivable. □

We now show that any proof in HOL-$\lambda\sigma$ can be transformed into a cut free proof, that is, a proof built without the rule *cut*. This result will be used twice, to show the equivalence between HOL-λ and HOL-$\lambda\sigma$ and to prove the completeness of the ENAR method for HOL-$\lambda\sigma$.

Proposition 5.4. (Cut elimination) The cut rule is redundant in HOL-$\lambda\sigma$.

Proof. Following the method developed in Dowek and Werner (1999), we construct a pre-model of the above rewrite system. A pre-model is a many-valued model whose truth values are reducibility candidates, that is, sets of proof-terms. Hence we will first define proof-terms, then reducibility candidates and pre-models and, finally, construct a pre-model for HOL-$\lambda\sigma$.

Proof-terms are inductively defined as follows:

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

Each proof-term construction corresponds to a natural deduction rule: terms of the form α express proofs built with the axiom rule, terms of the form $\lambda\alpha\ \pi$ and $(\pi\ \pi')$ express proofs built with the introduction and elimination rules of the implication, terms of the form (π, π') and $fst(\pi)$, $snd(\pi)$ express proofs built with the introduction and elimination rules of the conjunction, terms of the form $i(\pi), j(\pi)$ and $(\delta\ \pi_1\ \alpha\pi_2\ \beta\pi_3)$ express proofs built with the introduction and elimination rules of the disjunction, terms of the form $(botelim\ \pi)$ express proofs built with the elimination rule of the contradiction, terms of the form $\lambda x\ \pi$ and $(\pi\ t)$ express proofs built with the introduction and elimination rules of the universal quantifier and terms of the form (t, π) and $(exelim\ \pi\ x\alpha\pi')$ express proofs built with the introduction and elimination rules of the existential quantifier.

Reduction on these proof-terms is defined by the following rules, which eliminate cuts step by step:

$$(\lambda\alpha\ \pi_1\ \pi_2) \rhd \{\pi_2/\alpha\}\pi_1$$

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

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

$$(\delta\ i(\pi_1), \alpha\pi_2, \beta\pi_3) \rhd \{\pi_1/\alpha\}\pi_2$$

$$(\delta\ j(\pi_1), \alpha\pi_2, \beta\pi_3) \rhd \{\pi_1/\beta\}\pi_3$$

$$(\lambda x\ \pi\ t) \rhd \{t/x\}\pi$$

$$(exelim\ (t, \pi_1)\ \alpha x\pi_2) \rhd \{t/x, \pi_1/\alpha\}\pi_2$$

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

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

$$(exelim\ \pi_1\ x\alpha\pi_2) \rhd \pi_2.$$

We are now ready to define reducibility candidates. We recall that a proof-term is said to be *neutral* if it is a proof variable or an elimination (that is, 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. A set R of proof-terms 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 $\mathscr{C}$ for the set of all reducibility candidates.

A *pre-model* for a language $\mathscr{L}$ is given by:

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

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

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

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

— A proof π is an element of $|P(t_1, \ldots, t_n)|_\varphi$ if it is an element of $\overline{P}(|t_1|_\varphi, \ldots, |t_n|_\varphi)$.
— A proof π is element of $|A \Rightarrow B|_\varphi$ if it is strongly normalizable and if when it reduces to a proof of the form $\lambda\alpha\pi_1$, we have 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 if when it reduces to a proof of the form (π_1, π_2), we have 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 if when it reduces to a proof of the form $i(\pi_1)$ (respectively, $j(\pi_2)$), we have π_1 (respectively, π_2) is an element of $|A|_\varphi$ (respectively, $|B|_\varphi$).
— 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 if when it reduces to a proof of the form $\lambda x\ \pi_1$, we have for every term t of sort T (where T is the sort of x) and every element E of $\mathscr{M}_T$ that $\{t/x\}\pi_1$ 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 there exists an element E of $\mathscr{M}_T$ (where T is the sort of x) such that when π reduces to a proof of the form (t, π_1), then π_1 is an element of $|A|_{\varphi+(x,E)}$.

A pre-model is said to be a *pre-model of a congruence* $\equiv$ if when $A \equiv B$ we have for every assignment φ that $|A|_\varphi = |B|_\varphi$.

It is proved in Dowek and Werner (1999) that if a congruence $\equiv$ has a pre-model, then the cut rule is redundant in (intuitionistic) sequent calculus modulo $\equiv$.

To prove that the cut rule is redundant in sequent calculus modulo HOL-$\lambda\sigma$, we construct a pre-model of this theory. We let:

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

Then we interpret the symbols of the language as follows:

— $\overline{1_A^\Gamma}$ is the function mapping $a_1,\dots,a_n$ to a_1.
— $\overline{\alpha_{A\to B,A}^\Gamma}$ is the function mapping a and b to the function mapping $c_1,\dots,c_n$ to $a(c_1,\dots,c_n)(b(c_1,\dots,c_n))$.
— $\overline{\lambda_{A,B}^\Gamma}$ 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 the set of proofs π such that π is strongly normalizable and when π reduces to a proof of the form $\lambda\alpha\pi_1$, we have 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), we have 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 $i(\pi_1)$, we have π_1 is an element of a, and when π reduces to a proof of the form $j(\pi_2)$, we have 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 $\lambda\alpha\pi_1$, we have for every π' in a that $\{\pi'/\alpha\}\pi_1$ is strongly normalizable.
— $\overline{\dot\bot}$ if the set of strongly normalizable proofs.
— $\overline{\dot\forall_T}$ 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 $\lambda x\,\pi_1$, we have for every term t of sort T and every element E of $\mathscr{M}_T$ that $\{t/x\}\pi_1$ is an element of $(a\ E)$.
— $\overline{\dot\exists_T}$ is the function mapping a to the set of proofs π such that π is strongly normalizable and there exists an element E of $\mathscr{M}_T$ such that when π reduces to a proof of the form (t,π_1), then π_1 is an element of $(a\ E)$.
— $\bar\varepsilon$ is the identity function.

We check that the rules of $\lambda\sigma\mathscr{L}$ are valid in this pre-model, and hence $\lambda\sigma\mathscr{L}$ has a pre-model and the cut rule is redundant in sequent calculus modulo $\lambda\sigma\mathscr{L}$.

Following the technique introduced in Dowek and Werner (1999), we can lift the cut elimination result to the classical sequent calculus modulo $\lambda\sigma\mathscr{L}$. □

6. Embedding HOL-λ into HOL-λσ

We now want to prove that HOL-λσ is intentionally equivalent to the usual presentation of higher-order logic HOL-λ. First, we have to translate a λ-term, say a, into a λσ-term. To motivate the way the translation is done, we recall, following Dowek *et al.* (2000), that bound variables of a serve for reduction but the free variables serve only to be instantiated during a search proof process. In other words, the term a can be seen as a term with holes (called a context in Barendregt (1984)) filled by its free variables. So, bound variables are translated into de Bruijn numbers, letting the λσ-rules do reduction. But, free variables must be translated into variables to remain instantiable, so the translation of a is an open term of the first-order theory. However, as explained in Dowek *et al.* (2000), as capture has to be avoided, first-order substitution alone cannot perform the instantiations. We need to use a translation, called *pre-cooking*, which translates free variables on variables of the first-order theory, relocated by an appropriate $[\uparrow^n]$ operator according to the form of the context. Then, correctly instantiating free variables by a simple first-order substitution is recovered. Let us recall how pre-cooking is defined.

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

Definition 6.1. Let a be a λ-term. The pre-cooking of a is the λσ-term defined by $a_F = F(a, [\,])$ where $F(a, l)$ is defined using the list of variables 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.

We can now state our main theorem.

Theorem 6.1. If $p_1, \ldots, p_n, q_1, \ldots, q_m$ are propositions in HOL-λ, then $p_1, \ldots, p_n \vdash q_1, \ldots, q_m$ is provable in HOL-λ iff $\varepsilon(p_{1F}), \ldots, \varepsilon(p_{nF}) \vdash \varepsilon(q_{1F}), \ldots, \varepsilon(q_{nF})$ is provable in HOL-λσ.

The proof of this result relies on the following propositions. The first one recalls that pre-cooking is an homomorphism compatible with the respective substitution mechanisms and equalities.

Proposition 6.1. (Dowek *et al.* 2000)

— If t has the type T, then t_F has the sort $\vdash T$,
— $(\{a/x\}b)_F = \{x \mapsto a_F\}b_F$, where $\{x \mapsto a\}b$ denotes the first-order substitution of the (first-order) variable x by the term a in the term b (while $\{a/x\}b$ denotes the capture avoiding substitution of the λ-calculus).
— $a =_{\beta\eta} b$ in λ-calculus if and only if $a_F =_{\lambda\sigma} b_F$ in λσ-calculus.

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

Definition 6.2. An *F-term* is a $\lambda\sigma$-term containing only variables whose sort have empty contexts. An *F-proposition* is a proposition of the form $\varepsilon(P)$ where P is a F-term.

Proposition 6.2. If t is a $\lambda\sigma\mathscr{L}$-normal F-term well-typed in the empty context, 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 $\lambda\sigma\mathscr{L}$-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 as 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, and thus $s = \uparrow^n$. □

Proposition 6.3. Let $\Gamma \vdash \Delta$ be a sequent containing only F-propositions. Then, if this sequent has a proof, it also has a proof where all propositions are F-propositions and all the witnesses F-terms.

Proof. We use induction on the size of a cut free proof of $\Gamma \vdash \Delta$.

— If the last rule is an axiom, the result is obvious.
— If the last rule is a structural one, we apply the induction hypothesis to the subproofs.
— If the last rule is a left rule, we apply the induction hypothesis to the subproofs. The only non-trivial case is the left rule of the universal quantifier. The proof has the form

$$\dfrac{\dfrac{\pi}{\Gamma, R \vdash \Delta}}{\Gamma, \varepsilon(q) \vdash \Delta}\,(x, P, t)\ \forall\text{-l}$$

Where $\varepsilon(q) \equiv_{\lambda\sigma\mathscr{L}} \forall x\ P$ and $R \equiv_{\lambda\sigma\mathscr{L}} \{t/x\}P$. Hence $q \equiv_{\lambda\sigma\mathscr{L}} (\dot{\forall}\ p)$, $P \equiv_{\lambda\sigma\mathscr{L}} \varepsilon(p\ x)$ and $R \equiv_{\lambda\sigma\mathscr{L}} \varepsilon(p\ t)$. We use σ for 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 $\Gamma, \sigma R \vdash \Delta$, that is, $\Gamma, \varepsilon(p\ \sigma t) \vdash \Delta$. We apply the induction hypothesis to the proof $\sigma\pi$. Hence, there is a proof π' of $\Gamma, \varepsilon(p\ \sigma t) \vdash \Delta$ where all propositions are F-propositions and all the witnesses F-terms. We build the proof

$$\dfrac{\dfrac{\pi'}{\Gamma, \varepsilon(p\ \sigma t) \vdash \Delta}}{\Gamma, \varepsilon(q) \vdash \Delta}\,(x, \varepsilon(p\ x), \sigma t)\ \forall\text{-l}$$

— If the last rule is a right rule, we apply the induction hypothesis to the subproofs. The only non-trivial case is the right rule of the existential quantifier. We proceed as for the left rule of the universal quantifier. □

We can now give the proof of Theorem 6.1.

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

$$\dfrac{\dfrac{\pi}{\Gamma, q \vdash \Delta}}{\Gamma, p \vdash \Delta}\,(r, t)\ \dot{\forall}\text{-l}$$

where $p = (\dot{\forall}\ r)$ and $q = (r\ t)\downarrow$. Then $\varepsilon(p_F) = \varepsilon(\dot{\forall}\ r_F) \equiv_{\lambda\sigma\mathscr{L}} \forall x\ \varepsilon(r_F\ x)$. By the induction hypothesis, there is a proof π' of the sequent $\Gamma_F, \varepsilon(q_F) \vdash \Delta_F$. We build the proof

$$\dfrac{\dfrac{\pi'}{\Gamma_F, \varepsilon(q_F) \vdash \Delta_F}}{\Gamma_F, \varepsilon(p_F) \vdash \Delta_F}\,(x, \varepsilon(r_F\ x), t_F)\ \forall\text{-l}$$

Conversely, by Proposition 6.3, we can build a proof of

$$\varepsilon(p_{1F}), \ldots, \varepsilon(p_{nF}) \vdash \varepsilon(q_{1F}), \ldots, \varepsilon(q_{mF})$$

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 $p_1, \ldots, p_n \vdash q_1, \ldots, q_m$ in HOL-λ. As an example, we give the case of the left rule of the universal quantifier. The proof has the form

$$\dfrac{\dfrac{\pi}{\Gamma_F, \varepsilon(q_F) \vdash \Delta_F}}{\Gamma_F, \varepsilon(p_F) \vdash \Delta_F}\,(x, \varepsilon(r_F), t_F)\ \forall\text{-l}$$

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

$$\dfrac{\dfrac{\pi'}{\Gamma, q \vdash \Delta}}{\Gamma, p \vdash \Delta}\,(s, t)\ \forall\text{-l}$$

□

7. Skolemization in HOL-λσ

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 allows us 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 in a substitution if it is applied to at least n terms such that their free variables are not bound above in the term. For instance, the term $\lambda y\ (f^1\ x\ y)$ can be used in a substitution, while the terms f^1, $(F\ f^1)$ and $\lambda x\ (f^1\ x\ y)$ cannot (Miller's conditions (Miller 1983; Miller 1987)).

A further motivation for expressing higher-order logic as a first-order theory is to avoid this cumbersome rule by reusing the usual first-order skolemization rule. We show below that when we apply the first-order skolemization rule to HOL-C we get conditions on Skolem symbols that are variants of Miller's conditions. In HOL-$\lambda\sigma$ we get exactly Miller's conditions.

7.1. *Miller's conditions in HOL-λ*

The naive treatment of skolemization in higher-order logic, which skolemizes

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

as

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

introduces a constant f of type $T \to U$ (where T is the type of x and U that of y). But this skolemization rule 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 (Andrews 1972). 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 (Miller 1983; Miller 1987) has proposed an alternative skolemization rule that skolemizes a proposition of the form

$$\forall x_1\ \forall x_2\ \ldots\ \forall x_n\ \exists y\ (P\ x_1\ x_2\ \ldots\ x_n\ y)$$

into

$$\forall x_1\ \forall x_2\ \ldots\ \forall x_n\ (P\ x_1\ x_2\ \ldots\ x_n\ (f^n\ x_1\ x_2\ \ldots\ x_n)).$$

Two conditions are added to the terms substituted for variables:

— the symbol f^n can be used only when applied to at least n arguments (for example, $(f^1\ x)$ can be used in a substitution, but f^1 alone cannot).

— the variables free in the necessary arguments cannot be bound by a λ above in the term (for example, $\lambda x\ (f^1\ y)$ can be used in a substitution, but $\lambda x\ (f^1\ x)$ cannot).

Remark 7.1. As is usual in higher-order logic, $\forall x\ P$ is a notation for the term $\dot{\forall}\ (\lambda x\ P)$ where $\dot{\forall}$ is a constant. With such a convention, the skolemized proposition $\forall x\ (P\ x\ (f^1\ x))$ itself does not verify the second condition since x is bound by the external quantifier. However, this does not rule out this proposition because Miller's conditions do not apply to all terms and propositions, but only to the terms substituted for variables.

7.2. *Combinators*

The 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

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

is skolemized as

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

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 arithmetic) but $f(x)$ is. In this way we get Miller's first condition. As there is no notion of binding, the second condition vanishes in this presentation.

7.3. *HOL-λσ*

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

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

is skolemized as

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

Again f is a unary function symbol and hence we recover Miller's first condition, but its rank is now $(\Gamma \vdash T)\Delta \vdash U$, that is, it maps an argument of sort $\Gamma \vdash T$ into a term of sort $\Delta \vdash U$. The sort of the argument expresses exactly Miller's second condition, as it restricts the free variables in this term.

When the context associated to all variables is empty, the proposition

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

is skolemized as

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

where f has rank $(\vdash T) \vdash U$, which requires the argument of f to be well-typed in the empty context. For instance, the λ-term $\lambda x\ (f^1\ x)$, which violates Miller's second condition, is expressed by the term $\lambda(f(1))$, which is not well-typed, while the term $\lambda x\ (f^1\ y)$, which verifies Miller's second condition, is expressed by the term $\lambda(f(y))$, which is well-typed.

Notice that the restriction is simpler in this case as it applies uniformly to all the terms of the language, not only to the terms substituted for variables (see Remark 7.1). The proposition $A = \forall x\ \varepsilon(P\ x\ f(x))$ is well-formed since the variable x bound by the quantifier $\forall$ is a variable of first-order logic and not a de Bruijn index. But there exists no term t of type o such that $A \equiv_{\lambda\sigma\mathscr{L}} \varepsilon(t)$ since the only candidate would be $t = (\dot{\forall}\ \lambda(P\ 1\ f(1)))$, which is ill-formed.

$$\frac{\{A_1,\ldots,A_n,B_1,\ldots,B_m\}[E_1] \quad \{\neg C_1,\ldots,\neg C_p,D_1,\ldots,D_q\}[E_2]}{\{B_1,\ldots,B_m,D_1,\ldots,D_q\}[E_1\cup E_2\cup\{A_1\ldots=^?_{\mathscr{E}} A_n=^?_{\mathscr{E}} C_1\ldots=^?_{\mathscr{E}} C_p\}]}\ \textbf{Ext. Res.}$$

$$\frac{C[E]}{c\ell(C[r]_p)[E\cup\{C_{|p}=^?_{\mathscr{E}} l\}]}\ \textbf{Ext. Nar.}\quad \text{if } l\to r\in\mathscr{R} \text{ and } C_{|p} \text{ is not a variable}$$

Fig. 5. Extended narrowing and resolution (ENAR)

8. Automated theorem proving in HOL-$\lambda\sigma$

We are now able to wrap-up 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 ENAR method developed in Dowek *et al.* (1998) to search proofs in HOL-$\lambda\sigma$.

8.1. *The ENAR method*

The ENAR method applies to congruences described by class rewrite systems, that is, pairs composed of a rewrite system $\mathscr{R}$ rewriting atomic propositions to propositions and a set of equational axioms $\mathscr{E}$ equating terms with terms and defining a congruence denoted $=_{\mathscr{E}}$.

Compared with first-order resolution, the ENAR method first replaces unification by equational unification modulo $\mathscr{E}$. The unification problems are kept as constraints, written $t=^?_{\mathscr{E}} u$, and a clause C constrained by a set of equations E is written $C[E]$. Hence, we construct refutations with the **Extended Resolution** rule presented in Figure 5. Then, as $\mathscr{R}$ rewrites atomic propositions to non-atomic ones, we need another rule that instantiates, rewrites and puts in clausal form the result using the operator $c\ell$. This rule is called **Extended Narrowing** by analogy with the narrowing rule of equational unification.

Theorem 8.1. (Dowek *et al.* 1998) Let $\mathscr{R}\mathscr{E}$ be a confluent and weakly terminating class rewrite system such that the cut rule is redundant in sequent calculus modulo $\mathscr{R}\mathscr{E}$. Then, the sequent

$$A_1,\ldots,A_n\vdash B_1,\ldots,B_m$$

is provable in sequent calculus modulo if and only if from the constrained clauses

$$c\ell(\{\{A_1\},\ldots,\{A_n\},\{\neg B_1\},\ldots,\{\neg B_m\}\})[\varnothing]$$

we can derive the empty clause constrained by an $\mathscr{E}$-unifiable set of equations.

8.2. *Applying ENAR to HOL-$\lambda\sigma$*

In Dowek *et al.* (1998), we applied ENAR to a first-order presentation of higher-order logic using combinators. The system $\mathscr{E}$ contains the conversion rules of combinators while the system $\mathscr{R}$ contains the rules relating the connectors and quantifiers with their replication at the term level. We have shown that the **Extended Narrowing** rule specializes to the **Splitting** rule of higher-order resolution (Huet 1972; Huet 1973). Unfortunately,

equational unification modulo the conversion axioms of combinators is not higher-order unification.

If we apply this method to HOL-$\lambda\sigma$, we obtain another proof search method for higher-order logic. As shown in the previous sections, HOL-$\lambda\sigma$ fulfills the hypotheses of Theorem 8.1, so this method is complete. The **Extended Narrowing** rule still specializes to the **Splitting** rule of higher-order resolution, but the unification required is the unification modulo the system $\lambda\sigma$, which we have shown to be equivalent to higher-order unification in Dowek *et al.* (2000). Thus, the method obtained in this way simulates higher-order resolution step by step.

9. 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 higher-order resolution exactly. Hence we have shown in this way that 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, provided we take the correct first-order expression of higher-order logic and the correct proof search method.

Expressing higher-order resolution in a first-order framework allows us to clarify its features: higher-order unification, the splitting rule and higher-order skolemization. Higher-order unification is equational unification in an appropriate theory. The splitting rule is an instance of the extended narrowing rule introduced in Dowek *et al.* (1998), 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.

Since we stay in a first-order setting, we can first reuse optimizations of first-order theorem proving such as redundancy criteria and subsumption. Second, extending the method to equational higher-order resolution requires us only to add more reduction rules to the rewrite system $\lambda\sigma\mathscr{L}$, then narrowing provides an equational higher-order unification algorithm (Kirchner and Ringeissen 1997) and the proof search method is complete provided deduction modulo the extended theory verifies the cut elimination property.

Acknowledgements

We would like to thank the anonymous referees for their useful comments and suggestions.

References

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

Andrews, P. (1971) Resolution in type theory. *Journal of Symbolic Logic* **36** 414–432.

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

Andrews, P. (1986) *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof*, Academic Press.

Barendregt, H. P. (1984) *The Lambda-Calculus, its syntax and semantics*, Second edition, Studies in Logic and the Foundation of Mathematics, Elsevier Science.

Church, A. (1940) A formulation of the simple theory of types. *Journal of Symbolic Logic* **5** 56–68.

Curien, P.-L., Hardin, T. and Lévy, J.-J. (1996) Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM* **43** (2) 362–397.

Dowek, G., Hardin, T. and Kirchner, C. (1998) Theorem proving modulo. Rapport de Recherche 3400, Institut National de Recherche en Informatique et en Automatique. (To appear in *Journal of Automated Reasoning*.)

Dowek, G., Hardin, T. and Kirchner, C. (1999) HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic. In: Narendran, P. and Rusinowitch, M. (eds.) Rewriting Techniques and Applications. *Springer-Verlag Lecture Notes in Computer Science* **1631** 317–331.

Dowek, G., Hardin, T. and Kirchner, C. (2000) Higher-order unification via explicit substitutions. *Information and Computation* **157** 183–235.

Dowek, G., Hardin, T., Kirchner, C. and Pfenning, F. (1996) Unification via explicit substitutions: The case of higher-order patterns. In: Maher, M. (ed.) *Joint International Conference and Symposium on Programming Logic*, The MIT press 259–273.

Dowek, G. and Werner, B. (1999) Proof normalization modulo. In: Types for proofs and programs 98. *Springer-Verlag Lecture Notes in Computer Science* **1657** 62–77.

Girard, J.-Y. (1970) 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: Fenstad, J. E. (ed.) *Second Scandinavian Logic Symposium*, North-Holland.

Girard, J.-Y. (1972) *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur*, Ph. D. thesis, Paris VII.

Girard, J.-Y., Lafont, Y. and Taylor, P. (1989) *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science **7**, Cambridge University Press.

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

Hindley, J. (1964) *The Church–Rosser Property and a Result in Combinatory Logic*, Ph. D. thesis, University of Newcastle-upon-Tyne.

Huet, G. (1972) *Constrained Resolution: A Complete Method for Type Theory*, Ph. D. thesis, Case Western Reserve University.

Huet, G. (1973) A mechanization of type theory. In: *Proceedings of the Third International Joint Conference on Articifical Intelligence* 139–146.

Jouannaud, J.-P. and Kirchner, H. (1986) Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing* **15** (4) 1155–1194. (Preliminary version in Proceedings 11th ACM Symposium on Principles of Programming Languages, Salt Lake City (USA) 1984.)

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

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

Magnusson, L. (1994) *The implementation of ALF, a proof editor based on Martin-Löf monomorphic type theory with explicit substitution*, Doctoral thesis, Chalmers University of Technology and University of Göteborg.

Marché, C. (1994) Normalised rewriting and normalised completion. In: Abramsky, S. (ed.) *Proceedings 9th IEEE Symposium on Logic in Computer Science, Paris, France* 394–403.

Melliès, P.-A. (1995) Typed λ-calculi with explicit substitutions may not terminate. In: Typed Lambda Calculi and Applications. *Springer-Verlag Lecture Notes in Computer Science* **902** 328–334.

Miller, D. (1983) *Proofs in higher order logic*, Ph. D. thesis, Carnegie Mellon University.

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

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

Muñoz, C. (1997b) *Un calcul de substitutions pour la représentation de preuves partielles en théorie de types*, Thèse de doctorat, Université Paris 7.

Nadathur, G. and Wilson, D. S. (1990) A representation of lambda terms suitable for operations on their intensions. In: Wand, M. (ed.) *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming*, ACM Press 341–348.

Nadathur, G. and Wilson, D. S. (1998) A notation for lambda terms: A generalization of environments. *Theoretical Computer Science* **198** (1-2) 49–98.

Peterson, G. and Stickel, M. (1981) Complete sets of reductions for some equational theories. *Journal of the ACM* **28** 233–264.

Plotkin, G. (1972) Building-in equational theories. *Machine Intelligence* **7** 73–90.

Rosen, B. (1973) Tree manipulation systems and Church–Rosser theorems. *J. Assoc. Comput. Mach.* **20** 160–187.

Stickel, M. (1985) Automated deduction by theory resolution. *Journal of Automated Reasoning* **1** (4) 285–289.

Viry, P. (1995) Rewriting modulo a rewrite system. Technical report TR-20/95, Dipartimento di informatica, Università di Pisa.

Viry, P. (1998) Adventures in sequent calculus modulo equations. In: Kirchner, C. and Kirchner, H. (eds.) Proceedings of the 2nd International Workshop on Rewriting Logic and its Applications, WRLA'98, Pont-à-Mousson, France. *Electronic Notes in Theoretical Computer Science* **15**.