Proof Normalization for a First-order Formulation of Higher-order Logic

Gilles Dowek

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Proof normalization for a first-order formulation of higher-order logic

Gilles Dowek

No 3383

April 1998

THÈME 2

Rapport de recherche

N 0249-6399

Proof normalization for a first-order formulation of higher-order logic

Gilles Dowek [*]

Thème 2 — Génie logiciel
et calcul symbolique
Projet Coq

Rapport de recherche n ° 3383 — April 1998 — 21 pages

Abstract: We define a notion of cut and a proof reduction process for a class of theories, including all equational theories and a first-order formulation of higher-order logic. Proofs normalize for all equational theories. We show that the proof of the normalization theorem for the usual formulation of higher-order logic can be adapted to prove normalization for its first-order formulation. The "hard part" of the proof, that cannot be carried out in higher-order logic itself (the normalization of the system F-omega) is left unchanged. Thus, from the point of view of proof normalization, defining higher-order logic as a different logic or as a first-order theory does not matter. This result also explains a relation between the normalization of propositions and the normalization of proofs in equational theories and in higher-order logic: normalizing propositions does not eliminate cuts, but it transforms them.

Key-words: cut, proof normalization, higher-order logic, equational theory.

(Résumé : tsvp)

[*] Gilles.Dowek@inria.fr http://pauillac.inria.fr/~dowek

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Normalisation des démonstrations pour une présentation au premier ordre de la logique d'ordre supérieur

Résumé : On définit une notion de coupure et un processus de réduction des démonstrations pour un ensemble de théories comprenant toutes les théories équationnelles et une présentation au premier ordre de la logique d'ordre supérieur. Les démonstrations sont normalisables dans toutes les théories équationnelles. On montre que la démonstration du théorème de normalisation pour la formulation traditionnelle de la logique d'ordre supérieur peut être adaptée pour montrer la normalisation de sa formulation au premier ordre. La "partie difficile" de la démonstration qui ne peut pas s'exprimer dans la logique d'ordre supérieur elle-même (la normalisation du système F-oméga) reste identique. Du point de vue de la normalisation des démonstrations, définir la logique d'ordre supérieur comme une logique à part ou comme une théorie du premier ordre est donc indifférent. Ce résultat explique aussi une relation entre la normalisation des propositions et la normalisation des démonstrations dans les théories équationnelles et dans la logique d'ordre supérieur : normaliser les propositions n'élimine pas les coupures, mais les transforme.

Mots-clé : coupure, normalisation des démonstrations, logique d'ordre supérieur, théorie équationnelle.

It is well-known that higher-order logic can be formulated as a (many-sorted) first-order theory. Such a formulation permits to separate in a clear way the logic that describes the general rules of reasoning and a theory that describes the rules specific to the objects of the discourse (in this case, the sets and the functions).

From a more technical point of view, such a reduction permits to deduce Henkin's higher-order completeness theorem from Gödel's first-order completeness theorem (see for instance [4, 5]). It permits also to use well-known first-order proof search methods for higher-order logic. In particular, as this theory is some kind of extended equational theory, the powerful methods designed for equational theories can be used for higher-order logic. However most efficient first-order proof search methods rely on the proof normalization theorem and the fact that searching for normal proofs (or proofs containing a very restricted form of cuts) is complete. In this paper, we are concerned with proof normalization in the sense of Prawitz [12], i.e. in natural deduction.

The proof normalization theorem for higher-order logic cannot be deduced from the first-order one. Indeed, unlike completeness that is proved once for all the first-order theories, the proof normalization theorem needs a specific proof for each theory. The so called proof normalization theorem for first-order logic is only a proof normalization theorem for the empty theory in first-order logic. Other theories, such as equality or arithmetic, have their own notion of cut and their own proof normalization theorem.

We define in this paper a notion of cut for the first-order formulation of higher-order logic and we show that the proof of the normalization theorem for higher-order logic can be adapted to prove normalization for this theory. The "hard part" of the proof, that cannot be carried out in higher-order logic itself (the normalization of the system F_ω [9]) is left unchanged.

1 A first-order formulation of higher-order logic

Higher-order logic is an extension of first-order logic with function variables, predicate variables, function terms to be substituted to function variables and predicate terms to be substituted to predicate variables.

In its first-order formulation, if t is a function term and u_1, ..., u_n are terms we cannot write the application of t to u_1, ..., u_n as $t(u_1, ..., u_n)$, but we need to introduce a function symbol α_n (for *apply*) and write this term $\alpha_n(t, u_1, ..., u_n)$. In the same way, if t is a predicate term and u_1, ..., u_n are terms we cannot write the application of t to u_1, ..., u_n as $t(u_1, ..., u_n)$, but we need to introduce a predicate symbol $\in_n$ and write this proposition $\in_n (t, u_1, ..., u_n)$. When $n = 1$ we usually write $a \in A$ instead of $\in_1 (A, a)$. When $n = 0$, t is a zero-ary predicate term (i.e. a proposition term) and $\in_0 (t)$ is the corresponding proposition.

If we curry the functions and define the predicates as functions mapping objects to proposition terms, we only need a binary function symbols α_1, from now on written α, and a unary predicate $\in_0$, from now on written ε. The term $\alpha_n(t, u_1, ..., u_n)$ is now written $\alpha(..., \alpha(t, u_1)..., u_n)$ and the proposition $\in_n (t, u_1, ..., u_n)$ $\varepsilon(\alpha(..., \alpha(t, u_1)..., u_n))$. As usual, we write $(t\ u)$ for $\alpha(t, u)$ and $(t\ u_1\ ...\ u_n)$ for $(...(t\ u_1)...u_n)$.

In higher-order logic, base objects, predicates on base objects, predicates on predicates on base objects, ... are distinguished, and if P is, for instance, a predicate on base objects, we can apply it only to base objects: terms such as $(P\ P)$ are forbidden by the syntax. Thus, higher-order logic is a many-sorted first-order theory.

Definition 1.1 *(Many-sorted first-order logic)*

(See, for instance, [6, 7] for a detailed presentation) A many-sorted first-order language is given by

- *a denumerable collection $\mathcal{S}$ of* sorts*,*
- *for each sort T of $\mathcal{S}$, a denumerably infinite collection V_T of variables, such that V_T and V_U are disjoint when T and U are distinct,*
- *a denumerable collection of* function symbols*, to each function symbol f is associated an element of $\mathcal{S}^{n+1}$ $(n \geq 0)$ called its* rank *(when $n = 0$, such a symbol is also called an* individual symbol*),*
- *a denumerable collection of* predicate symbols*, to each predicate symbol P is associated an element of $\mathcal{S}^n$ $(n \geq 0)$ called its* rank.

Terms *of sort T are inductively defined by*

- *variables of V_T are terms of sort T,*
- *if f is a function symbol of rank $(T_1, ..., T_n, T_{n+1})$ and t_1, ..., t_n are terms of sort $T_1, ..., T_n$ then $f(t_1, ..., t_n)$ is a term of sort T_{n+1}.*

Propositions *are inductively defined by*

- *if P is a predicate symbol of rank $(T_1, ..., T_n)$ and t_1, ..., t_n are terms of sort $T_1, ..., T_n$ then $P(t_1, ..., t_n)$ is a proposition,*
- *$\bot$ (*falsehood*) is a proposition,*
- *if A is a proposition then $\neg A$ is a proposition,*
- *if A and B are propositions then $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are propositions,*
- *if A is a proposition and x a variable then $\forall x\ A$ and $\exists x\ A$ are propositions.*

Deduction rules *are the usual ones, with the restriction that a variable of sort T can only be substituted by a term of sort T.*

Definition 1.2 *(Many-sorted first-order logic with equality)*

A theory $\mathcal{T}$ in a language $\mathcal{L}$ in many-sorted first-order logic with equality is the theory in many-sorted first-order logic in the langage $\mathcal{L}$ extended by predicate symbols $=_T$ of rank (T, T) for each sort T and formed with the axioms of $\mathcal{T}$ and the axioms

$$\forall x\ x =_T x \qquad \textit{(Identity)}$$

$$\overline{\forall}\ (x =_T y \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y])) \qquad \textit{(Leibniz' scheme)}$$

where $\overline{\forall}\ P$ *is the universal closure of the proposition* P.

Higher-order logic is a theory in many-sorted first-order logic with equality whose sorts are called *simple types.*

Definition 1.3 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.*

As usual we write $T_1 \to ... \to T_n \to U$ for $T_1 \to (... \to (T_n \to U)...)$.

The symbol α is generalized to $\alpha_{T,U}$.

To construct functional terms and predicate terms we have the comprehension schemes.

$$\exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = t)$$

and

$$\exists E\ \forall x_1\ ...\ \forall x_n\ (\varepsilon(E\ x_1\ ...\ x_n) \Leftrightarrow P)$$

In fact, it is well-known that the first scheme is equivalent to the instances

$$\exists s\ \forall x\ \forall y\ \forall z\ (s\ x\ y\ z) = ((x\ z)\ (y\ z))$$

$$\exists k\ \forall x\ \forall y\ (k\ x\ y) = x$$

and the second is equivalent to the instances

$$\exists E\ \forall x\ \forall y\ (\varepsilon(E\ x\ y) \Leftrightarrow x =_T y)$$

$$\exists B\ (\varepsilon(B) \Leftrightarrow \bot)$$

$$\exists N\ \forall x\ (\varepsilon(N\ x) \Leftrightarrow \neg\varepsilon(x))$$

$$\exists C\ \forall x\ \forall y\ (\varepsilon(C\ x\ y) \Leftrightarrow (\varepsilon(x) \wedge \varepsilon(y)))$$

$$\exists D\ \forall x\ \forall y\ (\varepsilon(D\ x\ y) \Leftrightarrow (\varepsilon(x) \vee \varepsilon(y)))$$

$$\exists I\ \forall x\ \forall y\ (\varepsilon(I\ x\ y) \Leftrightarrow (\varepsilon(x) \Rightarrow \varepsilon(y)))$$

$$\exists E\ \forall x\ \forall y\ (\varepsilon(E\ x\ y) \Leftrightarrow (\varepsilon(x) \Leftrightarrow \varepsilon(y)))$$

$$\exists A\ \forall x\ (\varepsilon(A\ x) \Leftrightarrow \forall y\ \varepsilon(x\ y))$$

$$\exists E\ \forall x\ (\varepsilon(E\ x) \Leftrightarrow \exists y\ \varepsilon(x\ y))$$

To have a genuine notation for objects we skolemize these axioms, and introduce symbols $S_{T,U,V}$, $K_{T,U}$, $\dot{=}_T$, $\dot{\bot}$, $\dot{\neg}$, $\dot{\wedge}$, $\dot{\vee}$, $\dot{\Rightarrow}$, $\dot{\Leftrightarrow}$, $\dot{\forall}_T$, $\dot{\exists}_T$. Together with the symbols $\alpha_{T,U}$ and ε, these symbols form the language $\mathcal{L}$.

Definition 1.4 *The language $\mathcal{L}$ is the language in many-sorted first-order logic with equality sorted by simple types containing the individual symbols*

- $S_{T,U,V}$ *of sort* $(T \to U \to V) \to (T \to U) \to T \to V$,
- $K_{T,U}$ *of sort* $T \to U \to T$,
- $\dot{=}_T$ *of sort* $T \to T \to o$,
- $\dot{\bot}$ *of sort* o,
- $\dot{\neg}$ *of sort* $o \to o$,
- $\dot{\wedge}$, $\dot{\vee}$, $\dot{\Rightarrow}$, $\dot{\Leftrightarrow}$ *of sort* $o \to o \to o$,
- $\dot{\forall}_T$ *and* $\dot{\exists}_T$ *of sort* $(T \to o) \to o$,

the function symbols

- $\alpha_{T,U}$ *of rank* $(T \to U, T, U)$,

the predicate symbol

- ε *of rank* (o).

At last, we take the following instances of the skolemized comprehension scheme.

Definition 1.5 *The theory $\mathcal{H}$ is the theory in many-sorted logic with equality in the language $\mathcal{L}$ containing the axioms*

$$\forall x\ \forall y\ \forall z\ (S_{T,U,V}\ x\ y\ z) = ((x\ z)\ (y\ z))$$

$$\forall x\ \forall y\ (K_{T,U}\ x\ y) = x$$

$$\forall x\ \forall y\ (\varepsilon(\dot{=}_T\ x\ y) \Leftrightarrow (x =_T y))$$

$$\varepsilon(\dot{\bot}) \Leftrightarrow \bot$$

$$\forall x\ (\varepsilon(\dot{\neg}\ x) \Leftrightarrow (\neg\varepsilon(x)))$$

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

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

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

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

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

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

Remark In the classical case, many symbols in this language are redundant. The situation is less clear in the intuitionistic case.

Remark There are two notions of function that must not be confused. The individual symbol S, for instance, is a term and thus it has a sort. This term expresses an object of the theory that happens to be a function, thus its sort is a functional type. In contrast, the function symbol $\alpha_{T,U}$, for instance, is not a term and does not express an object of the theory, but a function mapping objects of the theory to objects of the theory. The sorts of the mapped objects are indicated by the rank of this symbol.

This distinction can be compared to that of set theory, where we are used to distinguish sets as objects of the theory and sets of objects of the theory.

Now we want to define a notion of cut and a proof reduction process associated to the theory $\mathcal{H}$.

2 Proof normalization for equational theories

2.1 First-order logic with equality

Definition 2.1 *An* equality cut *is a proof of the form*

$$\dfrac{\dfrac{\dfrac{\overline{\forall\ (x = y \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y]))}}{t = t \Rightarrow (P'[z \leftarrow t] \Rightarrow P'[z \leftarrow t])}\,\forall\text{-}elim \qquad \dfrac{\overline{\forall x\ x = x}}{t = t}\,\forall\text{-}elim}{P'[z \leftarrow t] \Rightarrow P'[z \leftarrow t]}\,\Rightarrow\text{-}elim \qquad \dfrac{\pi}{P'[z \leftarrow t]}}{P'[z \leftarrow t]}\,\Rightarrow\text{-}elim$$

where P' is an instance of P.

It reduces to the proof

$$\dfrac{\pi}{P'[z \leftarrow t]}$$

Proposition 2.1 *Proofs normalize in first-order logic with equality.*

2.2 Equational theories

Definition 2.2 *An* equational theory *is a theory whose axioms are universal closures of propositions of the form $t = u$.*

Definition 2.3 *Let $\mathcal{T}$ be an equational theory. An* elementary conversion step *in $\mathcal{T}$ relating a proposition $P'[z \leftarrow t]$ and $P'[z \leftarrow u]$ is a part of a proof of the form*

$$\dfrac{\dfrac{\dfrac{\overline{\overline{\overline{\forall}}\ (x = y \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y]))}}{t = u \Rightarrow (P'[z \leftarrow t] \Rightarrow P'[z \leftarrow u])}\ \forall\text{-}elim \qquad \dfrac{\rho}{t = u}}{P'[z \leftarrow t] \Rightarrow P'[z \leftarrow u]}\ \Rightarrow\text{-}elim \qquad P'[z \leftarrow t]}{P'[z \leftarrow u]}\ \Rightarrow\text{-}elim$$

Where ρ is either a proof of the form

$$\dfrac{\overline{\overline{\overline{\forall}} t_0 = u_0}}{t = u}\ \forall\text{-}elim$$

or a proof of the form

$$\dfrac{\dfrac{\dfrac{\overline{\forall x\ \forall y\ (x = y \Rightarrow (x = x \Rightarrow y = x))}}{t = u \Rightarrow (u = u \Rightarrow u = t)}\ \forall\text{-}elim \qquad \dfrac{\overline{\overline{\overline{\forall}} t_0 = u_0}}{t = u}\ \forall\text{-}elim}{u = u \Rightarrow u = t}\ \Rightarrow\text{-}elim \qquad \dfrac{\overline{\forall x\ x = x}}{u = u}\ \forall\text{-}elim}{u = t}\ \Rightarrow\text{-}elim$$

A conversion step *is a sequence of elementary conversion steps. We write such a conversion step relating two propositions P and Q*

$$\begin{array}{c} P \\ \boxed{} \\ Q \end{array}$$

Proposition 2.2 *From a proof in the theory $\mathcal{T}$, we can build a proof where axioms of $\mathcal{T}$ are used in conversion steps only.*

Proof We replace the axioms

$$\overline{\overline{\overline{\forall}}\ t = u}$$

by

$$\begin{array}{c} \dfrac{\overline{\forall x\ x = x}}{t = t}\ \forall\text{-elim} \\ \boxed{} \\ \dfrac{t = u}{\overline{\overline{\forall}}\ t = u}\ \forall\text{-intro} \end{array}$$

Proposition 2.3

- *There is a conversion step from P to P.*
- *From a conversion step from P to Q, we can build a conversion step from Q to P.*
- *From a conversion step from P to Q and a conversion step from Q to R, we can build a conversion step from P to R.*

Proof

- Take the empty conversion step.
- Take the reverse conversion step built by induction over the structure of the conversion step from P to Q.
- Take the concatenation of the conversion steps.

Proposition 2.4

- *If P and Q have the same toplevel connective or quantifier, then from a conversion step relating P and Q we can build a conversion step relating their toplevel subformulas.*
- *From a conversion step from P to Q, we can build a conversion step from $P[x \leftarrow t]$ to $Q[x \leftarrow t]$.*
- *From a conversion step relating $t = u$ and $t' = u'$ we can build a conversion step relating $P[x \leftarrow t]$ and $P[x \leftarrow t']$ and another relating $P[x \leftarrow u]$ and $P[x \leftarrow u']$.*

Proof By induction on the length of the conversion step.

Definition 2.4 *A* cut*, in natural deduction, in the theory $\mathcal{T}$ is an introduction rule followed by a conversion step and an elimination rule.*

Definition 2.5 *(Proof reduction)*

- *The cut*

$$\begin{array}{c} \dfrac{\dfrac{\pi_1}{P} \quad \dfrac{\pi_2}{Q}}{P \wedge Q}\,\wedge\textit{-intro} \\ \boxed{} \\ \dfrac{P' \wedge Q'}{P'}\,\wedge\textit{-elim} \end{array}$$

is transformed into the proof

$$\begin{array}{c} \dfrac{\pi_1}{P} \\ \boxed{} \\ P' \end{array}$$

The conversion step from P to P' is given by the first point of proposition 2.4. The case of the cuts using the right elimination rule is similar.

- *The cut*

$$\dfrac{\begin{array}{c}\dfrac{\dfrac{\pi_1}{P}}{P \vee Q}\ \vee\textit{-intro} \\ \boxed{} \\ P' \vee Q'\end{array} \qquad \begin{array}{c}\cancel{P'} \\ \dfrac{\pi_2}{R}\end{array} \qquad \begin{array}{c}\cancel{Q'} \\ \dfrac{\pi_3}{R}\end{array}}{R}\ \vee\textit{-elim}$$

is transformed into the proof

$$\begin{array}{c}\dfrac{\pi_1}{P} \\ \boxed{} \\ P' \\ \dfrac{\pi_2}{R}\end{array}$$

The conversion step from P to P' is given by the first point of proposition 2.4. The case of the cuts using the right introduction rule is similar.

- *The cut*

$$\dfrac{\begin{array}{c}\cancel{P} \\ \dfrac{\dfrac{\pi_1}{Q}}{P \Rightarrow Q}\ \Rightarrow\textit{-intro} \\ \boxed{} \\ P' \Rightarrow Q'\end{array} \qquad \dfrac{\pi_2}{P'}}{Q'}\ \Rightarrow\textit{-elim}$$

is transformed into the proof

$$\begin{array}{c}\dfrac{\pi_2}{P'} \\ \boxed{} \\ P \\ \dfrac{\pi_1}{Q} \\ \boxed{} \\ Q'\end{array}$$

The conversion step from P' to P and from Q to Q' are given by the first point of proposition 2.4 and proposition 2.3. The cases of the cuts on negation ($\neg$) and equivalence ($\Leftrightarrow$) are similar.

- *The cut*

$$\begin{array}{c} \dfrac{\dfrac{\pi}{P}}{\forall x\ P}\ \forall\text{-}\mathit{intro}\ (x \text{ not free in the hypotheses}) \\ \boxed{} \\ \dfrac{\forall x\ P'}{P'[x \leftarrow t]}\ \forall\text{-}\mathit{elim} \end{array}$$

is transformed into the proof

$$\begin{array}{c} \dfrac{\pi[x \leftarrow t]}{P[x \leftarrow t]} \\ \boxed{} \\ P'[x \leftarrow t] \end{array}$$

The conversion step from $P[x \leftarrow t]$ to $P'[x \leftarrow t]$ is given by the first and second point of proposition 2.4.

- *The cut*

$$\dfrac{\begin{array}{c} \dfrac{\dfrac{\pi_1}{P[x \leftarrow t]}}{\exists x\ P}\ \exists\text{-}\mathit{intro} \\ \boxed{} \\ \exists x\ P' \end{array} \qquad \dfrac{\begin{array}{c} \cancel{P'} \\ \pi_2 \end{array}}{Q}}{Q}\ \exists\text{-}\mathit{elim}\ (x \text{ not free in } Q)$$

is transformed into the proof

$$\begin{array}{c} \dfrac{\pi_1}{P[x \leftarrow t]} \\ \boxed{} \\ P'[x \leftarrow t] \\ \dfrac{\pi_2[x \leftarrow t]}{Q} \end{array}$$

The conversion step from $P[x \leftarrow t]$ to $P'[x \leftarrow t]$ is given by the first and second point of proposition 2.4.

- *The cut*

$$\dfrac{\dfrac{\dfrac{}{u = u' \Rightarrow P[z \leftarrow u] \Rightarrow P[z \leftarrow u']} \qquad \begin{array}{c} \dfrac{\dfrac{}{\forall x\ x = x}}{t = t}\ \forall\text{-}\mathit{elim} \\ \boxed{} \\ u = u' \end{array}}{P[z \leftarrow u] \Rightarrow P[z \leftarrow u']}\ \Rightarrow\text{-}\mathit{elim} \qquad \dfrac{\pi}{P[z \leftarrow u]}}{P[z \leftarrow u']}\ \Rightarrow\text{-}\mathit{elim}$$

is transformed into the proof

$$\begin{array}{c} \dfrac{\pi}{P[z \leftarrow u]} \\ \boxed{} \\ P[z \leftarrow u'] \end{array}$$

where the conversion step from $P[z \leftarrow u]$ to $P[z \leftarrow u']$ is built using the third point of proposition 2.4 and proposition 2.3.

Proposition 2.5 *If there is a conversion step from P to Q then P and Q have the same toplevel connective or quantifier (and if one is atomic, the other also).*

Proof By induction on the length of the conversion step.

Proposition 2.6 *Proofs normalize in the theory $\mathcal{T}$.*

Proof Let $\pi_1, \pi_2, ...$ be a proof reduction sequence in the theory $\mathcal{T}$. By proposition 2.4 and 2.5, the sequence $\pi'_1, \pi'_2, ...$ obtained by removing the conversion steps and replacing all the atomic propositions by the proposition $c = c$, where c is a constant, is a proof reduction sequence in first-order logic with equality. Thus it is finite.

2.3 Equivalence axioms

We may also include, in the theory $\mathcal{T}$, axioms of the form $\overline{\forall}\ (P \Leftrightarrow Q)$ where P and Q are atomic propositions. For instance Peano's fourth axiom

$$\forall x\ \forall y\ (S(x) = S(y) \Leftrightarrow x = y)$$

We restrict however to theories $\mathcal{T}$ such that if $t = t \Leftrightarrow u = u'$ or $u = u' \Leftrightarrow t = t$ is an instance of an axiom of $\mathcal{T}$ then either u and u' are the same term, or $u = u'$ is an instance of an axiom of $\mathcal{T}$ or $u' = u$ is.

The notion of conversion step is a bit more difficult to define, because we do not have an equivalent of Leibniz' scheme for equivalence. Thus, we inductively define a set L of proofs containing the proofs of the form

$$\dfrac{\overline{\overline{\overline{\forall}\ (P \Leftrightarrow Q)}}}{P' \Leftrightarrow Q'}\ \forall\text{-elim}$$

and closed by the following constructions: if

$$\dfrac{\pi}{P \Leftrightarrow Q}$$

is an element of L then

$$\dfrac{\dfrac{\dfrac{\dfrac{\pi}{P \Leftrightarrow Q} \quad \dfrac{\cancel{Q \wedge R}}{Q}}{P} \qquad \dfrac{\cancel{Q \wedge R}}{R}}{P \wedge R} \qquad \dfrac{\dfrac{\dfrac{\pi}{P \Leftrightarrow Q} \quad \dfrac{\cancel{P \wedge R}}{P}}{Q} \qquad \dfrac{\cancel{P \wedge R}}{R}}{Q \wedge R}}{(P \wedge R) \Leftrightarrow (Q \wedge R)}$$

is an element of L, and similar rules for the other connectives and quantifiers.

We extend the definition of elementary conversion steps to consider also parts of proofs of the form

$$\dfrac{\dfrac{\pi}{P \Leftrightarrow Q} \quad P}{Q}\ \Leftrightarrow\text{-elim}$$

and

$$\dfrac{\dfrac{\pi}{P \Leftrightarrow Q} \quad Q}{P}\ \Leftrightarrow\text{-elim}$$

where π is a proof of L.

Proposition 2.2 that every proof can be transformed into a proof where axioms are used in conversion steps only still holds, replacing the axioms

$$\dfrac{}{\overline{\forall}\ (P \Leftrightarrow Q)}$$

by

$$\dfrac{\dfrac{\begin{matrix}\cancel{P}\\ \boxed{}\\ Q\end{matrix} \qquad \begin{matrix}\cancel{Q}\\ \boxed{}\\ P\end{matrix}}{P \Leftrightarrow Q}\ \Leftrightarrow\text{-intro}}{\overline{\forall}\ (P \Leftrightarrow Q)}\ \forall\text{-intro}$$

Propositions 2.3 and 2.4 still hold. Thus, the cuts and the proof reduction process can be defined as in definition 2.4 and 2.5. Proposition 2.5 still holds, thus proof normalization can be proved as in proposition 2.6.

2.4 Plotkin-Andrews quotient

In this section, we give another characterization of the cuts associated to the equational theory $\mathcal{T}$.

Call $\mathcal{R}$ the equivalence relation on propositions defined by $P\ \mathcal{R}\ Q$ if and only if there is a conversion step from P to Q.

Proposition 2.7 *If $P\ \mathcal{R}\ Q$ and $\mathcal{T} \vdash P$ then $\mathcal{T} \vdash Q$.*

From proposition 2.3, $\mathcal{R}$ is an equivalence relation. Thus, we can consider the quotient $\mathcal{P}/\mathcal{R}$ of classes of propositions modulo $\mathcal{R}$ (Plotkin-Andrews quotient [1, 11]). We define the same deduction rules on classes of $\mathcal{P}/\mathcal{R}$ as on propositions of $\mathcal{P}$.

Remark Proof checking is decidable if $\mathcal{R}$ is decidable and provided we indicate the substituted term in the elimination rule of the universal quantifier and the introduction rule of the existential quantifier.

We have the following equivalence result.

Proposition 2.8 *Let $\overline{P}$ be the class of P in the quotient. We have $\mathcal{T} \vdash P$ if and only if $\vdash \overline{P}$.*

Proof By induction over proof structure we have $\mathcal{T} \vdash P$ if and only if $\overline{\mathcal{T}} \vdash \overline{P}$. Then in the quotient the axioms of $\mathcal{T}$ either have the form $t = t$ or $P \Leftrightarrow P$. Hence they are provable and $\mathcal{T} \vdash P$ if and only if $\vdash \overline{P}$.

Remark In the quotient, there are fewer axioms and proofs are simpler, thus proof search is more efficient. Unification must however be replaced by equational unification. Indeed, a unifier of two propositions P and Q is a substitution θ such that θP and θQ are equal in the quotient, i.e. such that $\theta P \; \mathcal{R} \; \theta Q$.

Remark In the quotient, conversion steps relate identical propositions and they can be removed: a cut in the quotient is just an introduction rule followed by an elimination rule. Thus, the notion of cut of definition 2.4 corresponds to the standard notion of cut (i.e. an introduction rule followed by an elimination rule) in the quotient.

Example Consider [11] the associativity axiom (A)

$$\forall x \; \forall y \; \forall z \; (x + y) + z = x + (y + z)$$

Two propositions P and Q are related by the relation $\mathcal{R}$ if one can be obtained from the other by a rearranging of brackets.

The associativity axiom is equivalent to

$$\forall x \; \forall y \; \forall z \; x + (y + z) = x + (y + z)$$

and thus, in the quotient, it is subsumed by the identity axiom

$$\forall x \; x = x$$

We have $\mathcal{T} \vdash P$ if and only if $\vdash \overline{P}$.

Call $t = (a + b) + c$ and $u = a + (b + c)$.

In the quotient, the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\overline{\forall}\ (x = y \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y]))}}{t = u \Rightarrow (P'[z \leftarrow t] \Rightarrow P'[z \leftarrow u])}\,\forall\text{-elim} \qquad \dfrac{\overline{A}}{t = u}\,\forall\text{-elim}}{P'[z \leftarrow t] \Rightarrow P'[z \leftarrow u]}\,\Rightarrow\text{-elim} \qquad \dfrac{\pi}{P'[z \leftarrow t]}}{P'[z \leftarrow u]}\,\Rightarrow\text{-elim}$$

can be simplified to

$$\frac{\pi}{P'[z \leftarrow u]}$$

Remark In some cases, there is a confluent and normalizing rewrite system rewriting propositions to equivalent propositions and such that $P \mathcal{R} Q$ if P and Q have the same normal form. For instance, in the example above, we have the rewrite system

$$(x+y)+z \rhd x+(y+z)$$

In this case, normal forms can be chosen as representative of their classes. Normalizing propositions does not eliminate cuts, but it transforms them removing conversion steps.

3 Proof normalization for the first-order formulation of higher-order logic

3.1 Normalizing propositions

We first define a rewrite system for terms and propositions of higher-order logic that will be useful in the following.

Definition 3.1 *Let $\rhd$ be the following rewrite system on terms and propositions of the theory $\mathcal{H}$ (actually, since the language of propositions contains binders (quantifiers), it is rather a combinatory reduction system [10]).*

$$(S_{T,U,V}\ x\ y\ z) \rhd ((x\ z)\ (y\ z))$$

$$(K_{T,U}\ x\ y) \rhd x$$

$$\varepsilon(\dot{=}_T\ x\ y) \rhd x =_T y$$

$$\varepsilon(\dot{\bot}) \rhd \bot$$

$$\varepsilon(\dot{\neg}\ x) \rhd \neg\varepsilon(x)$$

$$\varepsilon(\dot{\wedge}\ x\ y) \rhd \varepsilon(x) \wedge \varepsilon(y)$$

$$\varepsilon(\dot{\vee}\ x\ y) \rhd \varepsilon(x) \vee \varepsilon(y)$$

$$\varepsilon(\dot{\Rightarrow}\ x\ y) \rhd \varepsilon(x) \Rightarrow \varepsilon(y)$$

$$\varepsilon(\dot{\Leftrightarrow}\ x\ y) \rhd \varepsilon(x) \Leftrightarrow \varepsilon(y)$$

$$\varepsilon(\dot{\forall}_T\ x) \rhd \forall y\ \varepsilon(x\ y)$$

$$\varepsilon(\dot{\exists}_T\ x) \rhd \exists y\ \varepsilon(x\ y)$$

Proposition 3.1 *This rewrite system is confluent and strongly normalizing.*

Proof As this system is orthogonal, it is confluent [10].

To prove that it is strongly normalizing we define a translation of the terms and the propositions of the theory $\mathcal{H}$ into the typed combinatory language S, K. In each type T, we chose a variable z_T.

- $||x|| = z_T$,
- $||S_{T,U,V}|| = S_{T,U,V}$, $||K_{T,U}|| = K_{T,U}$,
- $||\dot{=}_T|| = (I_{T \to T \to o}\ z_{T \to T \to o})$, where $I_T = (S_{T,T,T}\ K_{T,T \to T}\ K_{T,T})$,
- $||\dot{\bot}|| = (I_o\ z_o)$,
- $||\dot{\neg}|| = (I_{o \to o}\ z_{o \to o})$,
- $||\dot{\wedge}|| = ||\dot{\vee}|| = ||\dot{\Rightarrow}|| = ||\dot{\Leftrightarrow}|| = (I_{o \to o \to o}\ z_{o \to o \to o})$,
- $||\dot{\forall}_T|| = ||\dot{\exists}_T|| = (S_{T \to o,T,o}\ I_{T \to o}\ (K_{T,T \to o}\ z_T))$,
- $||(t\ u)|| = (||t||\ ||u||)$,
- $||\varepsilon(t)|| = ||t||$,
- $||t =_T u|| = (z_{T \to T \to o}\ ||t||\ ||u||)$,
- $||\bot|| = z_o$
- $||\neg P|| = (z_{o \to o}||P||)$,
- $||P \wedge Q|| = ||P \vee Q|| = ||P \Rightarrow Q|| = ||P \Leftrightarrow Q|| = (z_{o \to o \to o}\ ||P||\ ||Q||)$,
- $||\forall x\ P|| = ||\exists x\ P|| = ||P||$.

We check that if P rewrites in one step to Q, then $||P||$ rewrites in at least one step to $||Q||$. Let $P_1, P_2, ...$ be a reduction sequence in the system above, the sequence $||P_1||, ||P_2||, ...$ is a reduction sequence in the typed combinatory language S, K, thus it is finite [14].

3.2 Proof reduction

The axioms of the theory $\mathcal{H}$ of definition 1.5 have the form $\overline{\forall}\ t = u$ or $\overline{\forall}\ (P \Leftrightarrow Q)$, but P and Q are not always atomic propositions. Conversion steps and the theory $\mathcal{R}$ can be defined as above. Notice that we have $P\ \mathcal{R}\ Q$ if and only if P and Q have the same normal form for the rewrite system above.

Proposition 2.2 that from a proof in the theory $\mathcal{H}$, we can build a proof where axioms of $\mathcal{H}$ are used in conversion steps only still holds and proposition 2.3 and 2.4 also, although the proof of the first point of proposition 2.4 is different.

Proposition 3.2 *If P and Q have the same toplevel connective or quantifier, then from a conversion step relating P and Q we can build a conversion step relating their toplevel subformulas.*

Proof From a conversion step relating the propositions P and Q we can build a conversion sequence from P to Q for the rewrite system of definition 3.1. Since this system is confluent, we can build reduction sequences from P and Q to a proposition R. This proposition R has the same toplevel connective or quantifier as P and Q and we can build a reduction sequence from the toplevel subformulas of P and Q to the toplevel subformulas of R. From these reduction sequences, we can build a conversion step relating the toplevel subformulas of P and Q.

Thus, the cuts and the proof reduction process can be defined as in definition 2.4 and 2.5. But as we shall see the proposition 2.5 does not hold and thus, the normalization proof of proposition 2.6 does not go through.

3.3 Normalization

In the usual formulation of higher-order logic, the substitution of a predicate or a proposition variable may increase the complexity of the proposition. For instance, substituting $P \Rightarrow P$ for X in

$$X \Rightarrow X$$

yields

$$(P \Rightarrow P) \Rightarrow (P \Rightarrow P)$$

Thus, proof normalization cannot be proved like in the empty theory of first-order logic, by using the fact that the complexity of cut propositions decreases.

This is also the case in the quotient, if we chose normal forms as representative of their classes, as substituting $(\Rightarrow p\ p)$ for x in the proposition

$$\varepsilon(x) \Rightarrow \varepsilon(x)$$

yields

$$(\varepsilon(p) \Rightarrow \varepsilon(p)) \Rightarrow (\varepsilon(p) \Rightarrow \varepsilon(p))$$

In the first-order formulation, predicate and proposition variables are just variables of sort $T_1 \rightarrow ... \rightarrow T_n \rightarrow o$ and substituting such a variable does not change the complexity of a proposition. For instance, substituting $(\Rightarrow p\ p)$ for x in the proposition

$$\varepsilon(x) \Rightarrow \varepsilon(x)$$

yields

$$\varepsilon(\Rightarrow p\ p) \Rightarrow \varepsilon(\Rightarrow p\ p)$$

which has the same complexity as $\varepsilon(x) \Rightarrow \varepsilon(x)$.

But, this complexity may be increased by a conversion step, for instance the proposition above can be transformed into

$$(\varepsilon(p) \Rightarrow \varepsilon(p)) \Rightarrow (\varepsilon(p) \Rightarrow \varepsilon(p))$$

Thus, because some axioms permit to transform atomic propositions into non-atomic ones, the proposition 2.5 does not hold for the theory $\mathcal{H}$ and the normalization proof of proposition 2.6 does not go through. However, we have the following normalization proof which is an adaptation of the proof of [9].

Proposition 3.3 *Proofs normalize in the theory $\mathcal{H}$.*

Proof We associate to each sort of the theory $\mathcal{H}$ a sort of the system F_ω [9] (see also [8]).

- $|o| = |\iota| = *$,
- $|T \to U| = |T| \to |U|$.

To each term of sort T we associate a type constructor of sort $|T|$ in F_ω

- $|S_{T,U,V}| = \lambda x : |T|\ \lambda y : |U|\ \lambda z : |V|\ ((x\ z)\ (y\ z))$,
- $|K_{T,U}| = \lambda x : |T|\ \lambda y : |U|\ x$,
- $|\dot{=}_T| = \lambda x : *\ \lambda y : *\ \forall P : |T| \to *\ ((P\ x) \Rightarrow (P\ y))$,
- $|\dot{\bot}| = \forall x : *\ x$,
- $|\dot{\neg}| = \lambda a : *\ (a \Rightarrow (\forall x : *\ x))$,
- $|\dot{\wedge}| = \lambda a : *\ \lambda b : *\ \forall x : *\ ((a \to b \to x) \to x)$.
- $|\dot{\vee}| = \lambda a : *\ \lambda b : *\ \forall x : *\ ((a \to x) \to (b \to x) \to x)$,
- $|\dot{\Rightarrow}| = \lambda a : *\ \lambda b : *\ (a \to b)$,
- $|\dot{\Leftrightarrow}| = \lambda a : *\ \lambda b : *\ (\forall x : *\ (((a \to b) \to (b \to a) \to x) \to x)$,
- $|\dot{\forall}_T| = \lambda x : |T| \to *\ \forall y : |T|\ (x\ y)$,
- $|\dot{\exists}_T| = \lambda x : |T| \to *\ \forall z : *\ ((\forall y : |T|((x\ y) \to z)) \to z)$,
- $|(t\ u)| = (|t|\ |u|)$.

To each proposition we associate a type in F_ω. Propositions of the form $\varepsilon(t)$ are translated like their arguments.

- $|\varepsilon(t)| = |t|$,

and the translation follows that of [9] for the other propositions, i.e.

- $|t =_T u| = \forall P : |T| \to * \; ((P\ |t|) \Rightarrow (P\ |u|))$,
- $|\bot| = \forall X : * \; X$,
- $|\neg P| = |P| \Rightarrow (\forall X : * \; X)$,
- $|P \wedge Q| = \forall X : * \; ((|P| \to |Q| \to X) \to X)$.
- $|P \vee Q| = \forall X : * \; ((|P| \to X) \to (|Q| \to X) \to X)$,
- $|P \Rightarrow Q| = |P| \to |Q|$,
- $|P \Leftrightarrow Q| = \forall X : * \; (((|P| \to |Q|) \to (|Q| \to |P|) \to X) \to X)$,
- $|\forall x\ P| = \forall x : |T|\ |P|$,
- $|\exists x\ P| = \forall X : * \; ((\forall x : |T|(|P| \to X)) \to X)$,

To each proof of a proposition P, we associate a term in F_ω of type $|P|$. To a proof of the form

$$\begin{array}{c} \pi \\ \hline P \\ \boxed{} \\ P' \end{array}$$

we associate the term $|\pi|$, and the translation follows that of [9] for the other proofs, e.g. to a proof of the form

$$\dfrac{\dfrac{\pi_1}{P \Rightarrow Q} \quad \dfrac{\pi_2}{P}}{Q} \Rightarrow\text{-elim}$$

we associate the term $(|\pi_1|\ |\pi_2|)$.

If a proof π contains a cut then the term $|\pi|$ contains a redex and eliminating the cut in π corresponds to reducing the redex in $|\pi|$.

Let $\pi_1, \pi_2, ...$ be a proof reduction sequence in the theory $\mathcal{H}$. The sequence $|\pi_1|, |\pi_2|, ...$ is a reduction sequence in F_ω. Thus, by the strong normalization theorem of F_ω [9], it is finite.

Remark Instead of mapping both ι and o to $*$, we could follow the closer the proof of [9] and drop the first-order terms, i.e. take $|o| = *$ and $|\iota|$ to be undefined then $|T \to U| = |T| \to |U|$ when both $|T|$ and $|U|$ are defined, is equal to $|U|$ when $|U|$ is defined but $|T|$ is not and is undefined otherwise.

Conclusion

We have defined a notion of cut (an introduction rule followed by a conversion step and an elimination rule) and a proof reduction process for a large class of theories, including all equational theories and a first-order formulation of higher-order logic $\mathcal{H}$. Although the proof reduction process is the same, the normalization proof is different for equational theories and the theory $\mathcal{H}$. The normalization for equational theories is proved by an elementary reduction to first-order logic with equality. By Gödel's second incompleteness theorem, there is no such reduction for the theory $\mathcal{H}$.

Plotkin-Andrews quotient permits to remove the conversion steps and gives another characterization of cuts. The notion of cut introduced here corresponds to the standard notion of cut (i.e. an introduction rule followed by an elimination rule) in the quotient. This explains a relation between the normalization of propositions and the normalization of proofs: normalizing of propositions does not eliminate cuts, but it transforms them by removing the conversion steps.

From the point of view of proof normalization, defining higher-order logic as a different logic or as a first-order theory does not matter because the "hard part" that cannot be carried out in higher-order logic itself (the normalization of F_ω) is the same in both cases.

Acknowledgements

The author thanks Peter Andrews who pointed out a mistake in a previous version of this paper.

References

[1] P.B. Andrews, Resolution in type theory, *The Journal of Symbolic Logic*, 36, 3 (1971) pp. 414-432.

[2] P.B. Andrews, An introduction to mathematical logic and type theory: to truth through proof, *Academic Press*, Orlando (1986).

[3] A. Church, A formulation of the simple theory of types, *The Journal of Symbolic Logic*, 5 (1940) pp. 56-68.

[4] M. Davis, Invited commentary to [13], *Proceedings of the International Federation for Information Processing Congress*, North-Holland (1968).

[5] G. Dowek, Collections, types and sets. *Mathematical Structures in Computer Science* (to appear).

[6] H.B. Enderton, A mathematical introduction to logic, *Academic Press*, New-York (1972).

[7] J. Gallier, Logic in computer science, *Harper and Row*, New-York (1986).

[8] J.H. Geuvers, M.J. Nederhof, A modular proof of strong normalization, *Journal of Functional Programming* 2, 1 (1991) pp. 155-189.

[9] J.Y. Girard, Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur, *Thèse d'État*, Université de Paris 7 (1972).

[10] J.W. Klop, V. van Oostrom, F. van Raamsdonk, Combinatory reduction systems: introduction and survey, *Theoretical Computer Science*, 121 (1993) pp. 279-308.

[11] G. Plotkin, Building-in equational theories, *Machine Intelligence*, 7 (1972) pp. 73-90

[12] D. Prawitz, Natural deduction. A proof-theoretical study. Almqvist & Wiksell (1965).

[13] J.A. Robinson, New directions in mechanical theorem proving, *Proceedings of the International Federation for Information Processing Congress*, North-Holland (1968).

[14] W.W. Tait, Intensional interpretation of functionals of finite type I, *Journal of Symbolic Logic*, 32, 2 (1967) pp. 198-212.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

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