Lambda-Calculus, Combinators and the Comprehension Scheme

Gilles Dowek

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Lambda-calculus, combinators and the comprehension scheme

Gilles Dowek

N° 2565

Juin 1995

PROGRAMME 2

Calcul symbolique,
programmation
et génie logiciel

ISSN 0249-6399

INRIA
ROCQUENCOURT

Lambda-calculus, combinators and the comprehension scheme

Gilles Dowek *

Programme 2 — Calcul symbolique, programmation et génie logiciel

Projet Coq

Rapport de recherche n ° 2565 — Juin 1995 — 25 pages

Abstract: The presentations of type theory based on a comprehension scheme and on λ-calculus are equivalent, both in the sense that the latter is a conservative extension of the former and that it can be encoded into it preserving provability. A similar result holds for set theory. A short version of this paper appeared in the proceedings of the conference *Typed Lambda-calculi and Applications (1995)*.

Key-words: lambda-calculus, combinators, comprehension scheme, type theory, set theory

(Résumé : tsvp)

* Gilles.Dowek@inria.fr

Unité de recherche INRIA Rocquencourt

Le lambda-calcul, les combinateurs et le schéma de compréhension

Résumé : Les présentations de la théorie des types basées sur un schéma de compréhension et sur le λ-calcul sont équivalentes, à la fois dans le sens que l'une est une extension conservative de l'autre et qu'elle peut s'y coder en préservant la propriété de prouvabilité. La théorie des ensembles vérifie une propriété similaire. Une version courte de cet article est parue dans les actes de la conférence *Typed Lambda-calculi and Applications (1995)*.

Mots-clé : lambda-calcul, combinateurs, schéma de compréhension, théorie des types, théorie des ensembles

Introduction

In the presentation of a theory we can either choose to give notations for objects and axioms expressing the properties of these objects, or to give axioms expressing the existence of objects verifying the desired properties. For instance, relations with a maximal element can either be defined by the language $\leq$, M and the axiom

$$\forall x\ (x \leq M)$$

or by the language $\leq$ and the axiom

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

From an existential formulation we can produce an explicit one by skolemizing the axioms [13, 2]. The skolemized theory is a conservative extension of the non skolemized one. When we have existence and also unicity axioms, we can translate the skolemized language into the non skolemized one preserving provability and thus the two theories are equivalent in a stronger way.

Church's type theory (also called higher order logic) [3, 2] can be presented in several ways. A first presentation uses an explicit notation for functions, i.e. a notation $\lambda x\ a$ (λ-calculus) and an axiom scheme (conversion)

$$\forall x\ ((\lambda x\ a)\ x) = a$$

Another merely states an axiom scheme expressing the existence of functions (comprehension)

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

For instance, in the first formulation, we have an explicit notation $(\lambda x\ x)$ for the identity function and an axiom $\forall x\ ((\lambda x\ x)\ x) = x$. In the second formulation, we only have an axiom $\exists f\ \forall x\ (f\ x) = x$. The equivalence of the two formulations is informally stated in [2]. In this paper we prove this statement.

When we skolemize the comprehension scheme, we get an explicit presentation of type theory. The language obtained this way is not λ-calculus, but rather a language based on combinators (in the sense of Curry [4]). Indeed, when we skolemize this scheme we introduce for each term a and for each sequence of variables $x_1, ..., x_n$ a primitive symbol $c_{x_1,...,x_n,a}$, also written $x_1, ..., x_n \mapsto a$ and the comprehension scheme becomes

$$\forall x_1\ ...\ \forall x_n\ (((x_1, ..., x_n \mapsto a)\ x_1\ ...\ x_n) = a)$$

which is roughly the conversion scheme [2]. There are however two essential differences between this language and λ-calculus. First, when we skolemize the comprehension scheme, we get symbols $x_1, ..., x_n \mapsto a$ only for terms a that do not contain further abstractions. Then, as abstractions are primitive symbols in the language of the skolemized comprehension scheme, there is nothing like the rule of λ-calculus that permits to substitute under abstractions with renaming.

Thus, we have in fact three presentations of type theory: the presentation with the comprehension scheme, the one with the skolemized comprehension scheme and the one with λ-calculus. We show that these three presentations are equivalent. A weak equivalence result is that each presentation is a conservative extension of the previous. In this case, we have also a stronger result, each presentation can be translated into the previous one preserving provability.

A similar equivalence result for second order logic is proved in [8]. There is however a difference between the second order case and the higher order one, as in the latter extensionality seems to play a central role.

There are a few choices in the formulation of the comprehension scheme. First, we may take the n-ary comprehension scheme

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

or only the unary one

$$\exists f\ \forall x\ ((f\ x) = a)$$

In λ-calculus functions are unary and n-ary functions are coded as unary ones by currification. But, we show that this cannot be done with the comprehension scheme: the n-ary comprehension scheme cannot be derived from the unary one.

Another choice concerns the free variables of a. In the proposition

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

we may decide that all the free variables of a have to be among $x_1, ..., x_n$. We get this way the *closed comprehension scheme*. We may also decide that some free variables of a may be omitted as arguments of f, in this case, we quantify universally these variables at the head of the axiom to get a closed axiom. We get this way the *open comprehension scheme*

$$\forall y_1\ ...\ \forall y_p\ \exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = a)$$

These two formulations of type theory are equivalent, but they lead to different languages when we skolemize them.

The last point concerns set theory. Like type theory, set theory can be presented either with existence axioms (Zermelo's axioms, or refinements) or with a language for objects including symbols $\mathcal{P}$, $\bigcup$ and $\{,\}$ where $\mathcal{P}(A)$ is the power set of A, $\bigcup(A)$ is the union of the elements of A (also written $\bigcup_{x \in A} x$) and $\{,\}(A, B)$ is the pair containing A and B (also written $\{A, B\}$) and a notation $\{x \in A \mid P\}$ for the subset of A of the elements verifying P. This notation is quite similar to λ-calculus, as it binds the variable x in the proposition P. When we skolemize the existence axioms, we do not get the language with the binder, but again these languages are equivalent.

Presentations of type theory and set theory with existence axioms, skolemized existence axioms or with binders are useful in different situations. The presentation with existence axioms or skolemized existence axioms are better suited to express these theories in a first order setting (set theory is a first order theory and type theory can be coded as a first order theory, see, for instance, [5]). But theories with an explicit notation (i.e. with skolemized existence axioms or binders) are better suited when we want to use these theories in practice, as a language for mathematics. It seems that the presentations with binders are easier to use than the ones with skolemized axioms, but this statement still needs to be justified.

1 Type theory

1.1 Type theory based on λ-calculus

Types are constructed from two base types: ι for the individuals and o for the propositions with a type constructor $\rightarrow$ for functions.

Definition 1.1 (Types)
Types are inductively defined by

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

There is an infinite number of variables of each type. There is an infinite number of primitive symbols $=_T$ of type $T \rightarrow T \rightarrow o$.

Definition 1.2 (Terms and propositions)
Terms of type T are inductively defined by

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

- $\top$ and $\bot$ are terms of type o (resp. *truth* and *falsehood*),
- if A is a term of type o then $\neg A$ is a term of type o,
- if A and B are terms of type o then $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are terms of type o,
- if A is a term of type o and x a variable of type T then $\forall x\ A$ and $\exists x\ A$ are terms of type o.

Propositions are terms of type o.
The terms of the form $\lambda x\ a$ are called *abstractions*.

Definition 1.3 (Substitution)

- $x[x \leftarrow b] = b$,
- $y[x \leftarrow b] = y$,
- $c[x \leftarrow b] = c$, if c is a primitive symbol,
- $(c\ d)[x \leftarrow b] = (c[x \leftarrow b]\ d[x \leftarrow b])$,
- $(\lambda y\ c)[x \leftarrow b] = \lambda z\ (c[y \leftarrow z][x \leftarrow b])$ *where z is a fresh variable, i.e. a variable not occurring in $\lambda y\ c$ or b,*
- $\top[x \leftarrow b] = \top$, $\bot[x \leftarrow b] = \bot$,
- $(\neg A)[x \leftarrow b] = \neg(A[x \leftarrow b])$,
- $(A * B)[x \leftarrow b] = (A[x \leftarrow b]) * (B[x \leftarrow b])$ for $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$,
- $(Qy\ A)[x \leftarrow b] = Qz\ (A[y \leftarrow z][x \leftarrow b])$ where z is a fresh variable, i.e. a variable not occurring in $Qx\ A$ or b, for $Q \in \{\forall, \exists\}$.

Because several choices are possible for the variable z, substitution is not defined on terms, but rather on classes of terms equivalent modulo bound variables renaming.

Definition 1.4 (Axioms)
Conversion:

$$\forall y_1\ ...\ \forall y_p\ \forall x\ (((\lambda x\ a)\ x) = a)\ (\beta)$$

Extensionality:

$$\forall f\ \forall g\ ((\forall x\ ((f\ x) = (g\ x))) \Rightarrow (f = g))$$
$$\forall P\ \forall Q\ ((P \Leftrightarrow Q) \Rightarrow (P = Q))$$

Equality:

$$\forall x\ (x = x)$$
$$\forall w_1\ ...\ \forall w_p\ \forall x\ \forall y\ ((x = y) \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y]))$$

Remark. In the axiom schemes, the free variables of a (resp. P) different from x (resp. z) are quantified universally at the head of the proposition. This way all axioms are closed.

Remark. In the conversion scheme

$$\forall y_1\ ...\ \forall y_p\ \forall x\ ((\lambda x\ a)\ x) = a$$

the variable x is bound twice (once by the quantifier and once by the λ). If we want to avoid this, we can reformulate the axiom, for instance, as

$$\forall y_1\ ...\ \forall y_p\ \forall z\ ((\lambda x\ a)\ z) = a[x \leftarrow z]$$

Deduction rules are the usual ones. We can take presentations based on natural deduction, sequent calculus or Frege-Hilbert systems. In this paper we use a natural deduction presentation (see, for instance, [6]).

1.2 Hyper-combinators

Definition 1.5 (Hyper-combinators)
The set of hyper-combinators is the smallest set of λ-terms such that

- variables are hyper-combinators,
- primitive symbols are hyper-combinators,
- if a and b are hyper-combinators then $(a\ b)$ is a hyper-combinator,
- *if a is a hyper-combinator, all the free variables of a are among $x_1, ..., x_n$ and no subterm of a is an abstraction, then $\lambda x_1\ ...\ \lambda x_n\ a$ is a hyper-combinator,*
- $\top$ and $\bot$ are hyper-combinators,
- if A is a hyper-combinator then $\neg A$ is a hyper-combinator,
- if A and B are hyper-combinators then $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ are hyper-combinators,
- if A is a hyper-combinator and x a variable then $\forall x\ A$ and $\exists x\ A$ are hyper-combinators.

Remark. When we relax the rules above to allow abstractions to occur in the body of abstractions, we get super-combinators [9]. Thus, hyper-combinators are super-combinators, but $\lambda f\ \lambda x\ (f\ ((\lambda y\ y)\ x))$ is a super-combinator which is not a hyper-combinator.

Proposition 1.1 *If a is a hyper-combinator and an abstraction then $a[x \leftarrow b] = a$.*

Proof. A hyper-combinator which is an abstraction is a closed term.

Proposition 1.2 *If a and b are hyper-combinators, then $a[x \leftarrow b]$ is a hyper-combinator.*

Proof. By induction over the structure of a.

Definition 1.6 (λ-lifting)
Let a be a λ-term, we define a hyper-combinator a' as follows

- $x' = x$, if x is a variable,
- $c' = c$, if c is a primitive symbol,
- $(b\ c)' = (b'\ c')$,
- if $a = (\lambda x_1\ ...\ \lambda x_n\ b)$ (b not an abstraction) then let $y_1, ..., y_p$ be the free variables of a, let b' be the translation of b, let $c_1, ..., c_q$ be the maximal subterms of b' which are abstractions (these terms are closed hyper-combinators), let b'' be the term obtained by replacing every c_i by a fresh variable z_i, we let $a' = ((\lambda y_1\ ...\ \lambda y_p\ \lambda z_1\ ...\ \lambda z_q\ \lambda x_1\ ...\ \lambda x_n\ b'')\ y_1\ ...\ y_p\ c_1\ ...\ c_q)$,
- $\top' = \top$, $\bot' = \bot$,
- $(\neg A)' = (\neg A')$,
- $(A \wedge B)' = (A' \wedge B')$, $(A \vee B)' = (A' \vee B')$, $(A \Rightarrow B)' = (A' \Rightarrow B')$, $(A \Leftrightarrow B)' = (A' \Leftrightarrow B')$,
- $(\forall x\ A)' = \forall x\ A'$, $(\exists x\ A)' = \exists x\ A'$.

Proposition 1.3 *For every λ-term a, $a = a'$ is a theorem of type theory.*

Proof. By induction on the structure of a. The second extensionality axiom is used to deduce $\forall x\ A = \forall x\ A'$ and $\exists x\ A = \exists x\ A'$ from $A = A'$.

Corollary 1.4 *A proposition P is provable in type theory if and only if P' is provable in type theory.*

Remark. Now, we want to restrict type theory in such a way that all the intermediate propositions appearing in proofs are hyper-combinators. It is not obvious that if P is provable in the λ-calculus based type theory then P' is provable in this restriction. For instance, the proposition

$$((\lambda f\ \lambda x\ (f\ x))\ f\ x) = (f\ x)$$

is provable in the λ-calculus based theory: using twice the scheme β we prove that $((\lambda f\ \lambda x\ (f\ x))\ f\ x)$ is equal to $((\lambda x\ (f\ x))\ x)$ and then to $(f\ x)$. Since this proposition is a hyper-combinator, it is its own translation, but the proof above does not go through in the restriction, as the intermediate term $((\lambda x\ (f\ x))\ x)$ is not a hyper-combinator. Thus, we replace the scheme β by the scheme $\beta*$

$$\forall y_1\ ...\ \forall y_p\ \forall x_1\ ...\ \forall x_n\ ((\lambda x_1\ ...\ \lambda x_n\ a)\ x_1\ ...\ x_n) = a\ (a \text{ not an abstraction})\ (\beta*)$$

In type theory based on λ-calculus, the theory obtained by replacing β by $\beta*$ is obviously equivalent to the initial one, but it seems to be more powerful when the theory is restricted to hyper-combinators, as using this scheme we can prove the proposition

$$((\lambda f\ \lambda x\ (f\ x))\ f\ x) = (f\ x)$$

Definition 1.7 The *hyper-combinators based type theory* is the restriction of the λ-calculus based type theory such that all the intermediate propositions appearing in proofs are hyper-combinators and the scheme β is replaced by the scheme $\beta*$.

Remark. Even with the scheme $\beta*$, it is still not obvious that every proof in the λ-calculus based type theory can be translated as a proof of P' in the hyper-combinators based type theory. Consider for instance the terms

$$a = ((\lambda x\ \lambda y\ \lambda z\ x)\ w\ w) \qquad b = ((\lambda x\ \lambda y\ \lambda z\ y)\ w\ w)$$

The proposition $a = b$ is a theorem in the λ-calculus based type theory, as both a and b are provably equal to $\lambda z\ w$. But this proof does not go through in the hyper-combinators based type theory. In this theory, we need to use the extensionality axiom, then we are reduced to prove that for every u, $(a\ u) = (b\ u)$, i.e.

$$((\lambda x\ \lambda y\ \lambda z\ x)\ w\ w\ u) = ((\lambda x\ \lambda y\ \lambda z\ y)\ w\ w\ u)$$

and these terms are both provably equal to the term w.

Proposition 1.5 *If A is an axiom of the λ-calculus based type theory then A' is provable in the hyper-combinators based type theory.*

Proof.

- If A is an instance of the conversion scheme β

  $$\forall z_1\ ...\ \forall z_p\ \forall x\ ((\lambda x\ a)\ x) = a$$

  then we have to prove

  $$((\lambda x\ a)'\ x) = a'$$

Let us write $a = \lambda y_1 \ ... \ \lambda y_n \ b$ where b is not an abstraction. We have to prove

$$((\lambda x \ \lambda y_1 \ ... \ \lambda y_n \ b)' \ x) = (\lambda y_1 \ ... \ \lambda y_n \ b)'$$

By extensionality, we are reduced to prove

$$((\lambda x \ \lambda y_1 \ ... \ \lambda y_n \ b)' \ x \ y_1 \ ... \ y_n) = ((\lambda y_1 \ ... \ \lambda y_n \ b)' \ y_1 \ ... \ y_n)$$

and these terms are both provably equal to b'.

- If A has the form
$$\forall w_1 \ ... \ \forall w_p \ \forall x \ \forall y \ (x = y) \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y])$$
then A' is
$$\forall w_1 \ ... \ \forall w_p \ \forall x \ \forall y \ (x = y) \Rightarrow (P'[z \leftarrow x] \Rightarrow P'[z \leftarrow y])$$
which is an axiom.
- Otherwise the axiom is its own translation.

Proposition 1.6 *Let a and b be terms, the proposition $(a[x \leftarrow b])' = a'[x \leftarrow b']$ is a theorem of the hypercombinators based type theory.*

Proof. By induction over the structure of a.

- If a is a variable or a primitive symbol then the result is obvious.
- If $a = (c \ d)$ then we apply the induction hypothesis.
- If $a = \lambda y_1 \ ... \ \lambda y_n \ c$ (c not an abstraction) then if x is not free in a then the result is trivial, otherwise by extensionality we are reduced to prove
$$((a[x \leftarrow b])' \ y_1 \ ... \ y_n) = (a'[x \leftarrow b'] \ y_1 \ ... \ y_n)$$
We first show that the terms $(a'[x \leftarrow b'] \ y_1 \ ... \ y_n)$ and $c'[x \leftarrow b']$ are provably equal. Let $x, u_1, ..., u_p$ be the free variables of a. Let $e_1, ..., e_q$ the maximal abstractions in the term c', $v_1, ..., v_q$ be fresh variables and c'' the term c' where the terms $e_1, ..., e_q$ are replaced by the variables $v_1, ..., v_q$. We have
$$a' = ((\lambda x \ \lambda u_1 \ ... \ \lambda u_p \ \lambda v_1 \ ... \ \lambda v_q \ \lambda y_1 \ ... \ \lambda y_n \ c'') \ x \ u_1 \ ... \ u_p \ e_1 \ ... \ e_q)$$
$$(a'[x \leftarrow b'] \ y_1 \ ... \ y_n) = ((\lambda x \ \lambda u_1 \ ... \ \lambda u_p \ \lambda v_1 \ ... \ \lambda v_q \ \lambda y_1 \ ... \ \lambda y_n \ c'') \ b' \ u_1 \ ... \ u_p \ e_1 \ ... \ e_q \ y_1 \ ... \ y_n)$$
And this term is provably equal to
$$c''[x \leftarrow b', v_1 \leftarrow e_1, ..., v_q \leftarrow e_q] = c''[v_1 \leftarrow e_1, ..., v_q \leftarrow e_q][x \leftarrow b'] = c'[x \leftarrow b']$$
Then, we show that the terms $((a[x \leftarrow b])' \ y_1 \ ... \ y_n)$ and $(c[x \leftarrow b])'$ are provably equal. Let $\lambda z_1 \ ... \ \lambda z_m \ d$ (d not an abstraction) be the term $c[x \leftarrow b]$. We need to prove
$$((\lambda y_1 \ ... \ \lambda y_n \ \lambda z_1 \ ... \ \lambda z_m \ d)' \ y_1 \ ... \ y_n) = (\lambda z_1 \ ... \ \lambda z_m \ d)'$$
By extensionality, we are reduced to prove
$$((\lambda y_1 \ ... \ \lambda y_n \ \lambda z_1 \ ... \ \lambda z_m \ d)' \ y_1 \ ... \ y_n \ z_1 \ ... \ z_m) = ((\lambda z_1 \ ... \ \lambda z_m \ d)' \ z_1 \ ... \ z_m)$$
and these terms are both provably equal to d'.
At last, by induction hypothesis, the terms $c'[x \leftarrow b']$ and $(c[x \leftarrow b])'$ are provably equal. Thus, $(a'[x \leftarrow b'] \ y_1 \ ... \ y_n)$ and $((a[x \leftarrow b])' \ y_1 \ ... \ y_n)$ are provably equal.

- If $a = \top$, $a = \bot$, $a = \neg A$, $a = A \wedge B$, $a = A \vee B$, $a = A \Rightarrow B$, $a = A \Leftrightarrow B$, we apply the induction hypothesis.
- If $a = Qy\ A$ ($Q \in \{\forall, \exists\}$), by induction hypothesis, we have

$$A'[x \leftarrow b'] = (A[x \leftarrow b])'$$

Thus

$$A'[x \leftarrow b'] \Leftrightarrow (A[x \leftarrow b])'$$
$$\forall y\ (A'[x \leftarrow b'] \Leftrightarrow (A[x \leftarrow b])')$$
$$(Qy\ (A'[x \leftarrow b'])) \Leftrightarrow (Qy\ (A[x \leftarrow b])')$$
$$((Qy\ A)'[x \leftarrow b']) \Leftrightarrow ((Qy\ A)[x \leftarrow b])'$$

By the second extensionality axiom, we get

$$((Qy\ A)'[x \leftarrow b']) = ((Qy\ A)[x \leftarrow b])'$$

Proposition 1.7 *A proposition P is provable in the λ-calculus based type theory if and only if P' is provable in the hyper-combinators based type theory.*

Proof. Assume P is provable in the presentation of type theory based on λ-calculus. By induction on the structure of the proof of P we show that P' is provable in the hyper-combinators based type theory. Most cases are trivial, we use the proposition 1.5 for the axiom rule and the proposition 1.6 for the elimination rule of the universal quantifier and the introduction rule of the existential quantifier.

Conversely, if P' is provable in the hyper-combinators based type theory, then P' is provable in the λ-calculus based type theory and as $P = P'$ is provable in this theory, so is P.

Proposition 1.8 *Type theory based on λ-calculus is a conservative extension of type theory based on hyper-combinators.*

Proof. If P is a hyper-combinator, then $P' = P$. Thus P is provable in type theory based on λ-calculus if and only if it is provable in type theory based on hyper-combinators.

Proposition 1.9 *Type theory based on λ-calculus and on hyper-combinators are equivalent both in the sense that the former is a conservative extension of the latter and in the sense that it can be translated into it preserving provability.*

1.3 The comprehension scheme

In type theory with the comprehension scheme, we do not have the notation $\lambda x\ a$ any more, thus we have the following definitions.

Definition 1.8 (Type theory with the open comprehension scheme)

Types are defined as in Definition 1.1, *terms* and *propositions* are defined as in Definition 1.2 without the clause 4 and *substitution* is defined as in Definition 1.3 without the clause 5. Equality and extensionality axioms are the same, but we drop the conversion scheme and add the comprehension scheme

$$\forall y_1\ ...\ \forall y_p\ \exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = a)$$

where f is not free in a and $y_1, ..., y_p$ are the free variables of a that are not among $x_1, ..., x_n$. Deduction rules are the same as in the λ-calculus based presentation.

Definition 1.9 (Type theory with the closed comprehension scheme)
Type theory with the closed comprehension scheme is the restriction of type theory with the open comprehension scheme, where we take only the instances of the scheme such that all the free variables of a are among $x_1, ..., x_n$. In such an instance, $p = 0$ and the comprehension scheme is rephrased

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

Proposition 1.10 *Type theory with the open or the closed comprehension scheme are equivalent.*

Proof. The open scheme trivially implies the closed one. Let us show that conversely the closed scheme implies the open one. Let a be a term, and $x_1, ..., x_n$ be variables, let $y_1, ..., y_p$ be the free variables of a that are not among $x_1, ..., x_n$. Let us prove, in type theory with the closed comprehension scheme, the proposition

$$\forall y_1\ ...\ \forall y_p\ \exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = a)$$

We have the axiom

$$\exists g\ \forall y_1\ ...\ \forall y_p\ \forall x_1\ ...\ \forall x_n\ ((g\ y_1\ ...\ y_p\ x_1\ ...\ x_n) = a)$$

From the hypothesis

$$\forall y_1\ ...\ \forall y_p\ \forall x_1\ ...\ \forall x_n\ ((g\ y_1\ ...\ y_p\ x_1\ ...\ x_n) = a)$$

we deduce

$$\forall x_1\ ...\ \forall x_n\ ((g\ y_1\ ...\ y_p\ x_1\ ...\ x_n) = a)$$
$$\exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = a)$$
$$\forall y_1\ ...\ \forall y_p\ \exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = a)$$

Using the elimination rule of the existential quantifier we get a proof of

$$\forall y_1\ ...\ \forall y_p\ \exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = a)$$

with no hypotheses.

1.4 The skolemized comprehension scheme

When we skolemize the closed comprehension scheme, we introduce an infinite number of primitive symbols $c_{x_1,...,x_n,a}$ (also written $x_1, ..., x_n \mapsto a$) where all the free variables of a are among $x_1, ..., x_n$ and a does not contain symbols of the form $y_1, ..., y_p \mapsto b$. We get the axiom

$$\forall x_1\ ...\ \forall x_n\ (((x_1, ..., x_n \mapsto a)\ x_1\ ...\ x_n) = a)$$

Remark. When we define the language of this theory, we first define the set of terms without Skolem symbols, then for each such term a and sequence of variables $x_1, ..., x_n$ where all the free variables of a are among $x_1, ..., x_n$ we introduce a Skolem symbol $x_1, ..., x_n \mapsto a$, at last we define the full set of terms.

Proposition 1.11 *If a proposition P contains no symbol $x_1, ..., x_n \mapsto a$, then P is provable in the theory of the comprehension scheme if and only if it is provable in the theory of the skolemized comprehension scheme. Thus, the skolemized theory is a conservative extension of the non skolemized one.*

Proof. Consider a proof of P in type theory with the skolemized comprehension scheme, we replace every Skolem symbol $x_{i,1}, ..., x_{i,n_i} \mapsto a_i$ by a variable f_i then we eliminate all the hypotheses of the form

$$\forall x_{i,1}\ ...\ \forall x_{i,n_i}\ ((f_i\ x_{i,1}\ ...\ x_{i,n_i}) = a_i)$$

with the elimination rule of the existential quantifier and the comprehension scheme. We get this way a proof of P in type theory with the comprehension scheme.

Conversely, we can prove the comprehension scheme in type theory with the skolemized comprehension scheme. Thus, if P has a proof in type theory with the comprehension scheme, it has also a proof in the theory with the skolemized comprehension scheme.

Definition 1.10 Let P be a proposition in the language of the skolemized comprehension scheme, we define the proposition P° in the language of the (non skolemized) comprehension scheme as follows. We replace in P each symbol $x_{i,1}, ..., x_{i,n_i} \mapsto a_i$ by a variable f_i, we get a proposition P_1 and we take

$$P^\circ = \exists f_1 \ ... \ \exists f_p \ ((\forall x_{1,1} \ ... \ \forall x_{1,n_1} \ (f_1 \ x_{1,1} \ ... \ x_{1,n_1}) = a_1) \wedge ... \wedge (\forall x_{p,1} \ ... \ \forall x_{p,n_p} \ (f_p \ x_{p,1} \ ... \ x_{p,n_p}) = a_p) \wedge P_1)$$

Proposition 1.12 *Let P be a proposition in the language of the skolemized theory. The proposition $P \Leftrightarrow P^\circ$ is provable in the skolemized theory.*

Proof. Assume P, then P° is provable by taking $x_{i,1}, ..., x_{i,n_i} \mapsto a_i$ for f_i. Assume conversely P°, then from the hypothesis

$$(\forall x_{1,1} \ ... \ \forall x_{1,n_1} \ (f_1 \ x_{1,1} \ ... \ x_{1,n_1}) = a_1) \wedge ... \wedge (\forall x_{p,1} \ ... \ \forall x_{p,n_p} \ (f_p \ x_{p,1} \ ... \ x_{p,n_p}) = a_p) \wedge P_1$$

using the extensionality axiom we can prove

$$f_1 = x_{1,1}, ..., x_{1,n_1} \mapsto a_1$$

$$...$$

$$f_p = x_{p,1}, ..., x_{p,n_p} \mapsto a_p$$

thus, using the axioms of equality, we can deduce the proposition P. Then using the elimination rule of the existential quantifier we can prove the proposition P from P°.

Corollary 1.13 *The proposition P is provable in the skolemized theory if and only if P° is provable in the non skolemized theory.*

Proposition 1.14 *The theory of the comprehension scheme and the skolemized comprehension scheme are equivalent both in the sense that the latter is a conservative extension of the former and in the sense that it can be translated into it preserving provability.*

Remark. Skolemization must be used with care in type theory: some formulations are always sound while others are sound only when the axiom of choice is assumed in the theory. Here we use only external skolemization, i.e. we replace axioms of the form $\exists x \ P$ by the axioms $P[x \leftarrow c]$. As shown above, this rule is always sound.

1.5 The skolemized comprehension scheme and hyper-combinators

Now, we can show that there is a simple one-to-one translation from type theory based on hyper-combinators and on the skolemized closed comprehension scheme. In both cases $\lambda x_1 \ ... \ \lambda x_n \ a$ and $x_1, ..., x_n \mapsto a$ are closed terms and a does not contain further abstractions. Thus, the only difference between the hyper-combinators and the language of the skolemized comprehension scheme is that the former provides a syntactical rule to construct abstractions, while the latter provides a new primitive symbol for each abstraction.

Definition 1.11 Let a be a term in the hyper-combinators based type theory, the translation a^+ of a is a term in type theory with the skolemized closed comprehension scheme defined as follows

- $x^+ = x$,
- $c^+ = c$,
- $(a \ b)^+ = (a^+ \ b^+)$,
- $(\lambda x_1 \ ... \ \lambda x_n \ a)^+ = x_1, ..., x_n \mapsto a^+$ (a not an abstraction),

- $\top^+ = \top$, $\bot^+ = \bot$,
- $(\neg A)^+ = \neg A^+$,
- $(A \wedge B)^+ = A^+ \wedge B^+$, $(A \vee B)^+ = A^+ \vee B^+$, $(A \Rightarrow B)^+ = A^+ \Rightarrow B^+$, $(A \Leftrightarrow B)^+ = A^+ \Leftrightarrow B^+$,
- $(\forall x\ A)^+ = \forall x\ A^+$, $(\exists x\ A)^+ = \exists x\ A^+$.

This translation is obviously one to one, we write $^-$ for its converse.

Proposition 1.15 *Let a and b be two hyper-combinators, the proposition*

$$(a[x \leftarrow b])^+ = a^+[x \leftarrow b^+]$$

is provable in type theory with the skolemized closed comprehension scheme. Conversely, let a and b be two terms in the language of the skolemized closed comprehension scheme, the proposition

$$(a[x \leftarrow b])^- = a^-[x \leftarrow b^-]$$

is provable in the hyper-combinators based type theory.

Proof. By induction over the structure of a. The second extensionality axiom is used for the case of quantifiers.

Proposition 1.16 *A proposition P is provable in type theory with hyper-combinators if and only if P^+ is provable in type theory with the skolemized comprehension scheme.*

Proof. By induction over proof structure.

Theorem 1.1 *The three presentations of type theory (based on the comprehension scheme, on the skolemized comprehension scheme (i.e. hyper-combinators) and on λ-calculus) are equivalent, both in the sense that each one is a conservative extension of the previous and that each one can be encoded into the previous preserving provability.*

Remark. The equivalence result above holds both for classical type theory and intuitionistic type theory.

2 Independence of the binary comprehension scheme

In λ-calculus repeated application of the λ-rule permits to form n-ary functions. For instance from the term x we can form the term $\lambda y\ x$ and then $\lambda x\ \lambda y\ x$. In contrast, it is not possible to iterate the use of the unary comprehension scheme to build n-ary functions. Indeed, we have the following instances of the comprehension scheme

$$\forall x\ \exists f\ \forall y\ (f\ y) = x \qquad \forall f\ \exists g\ \forall x\ (g\ x) = f$$

Although using the hypotheses $\forall y\ (f\ y) = x$ and $\forall x\ (g\ x) = f$ we can derive the proposition

$$\forall y\ (g\ x\ y) = x$$

we cannot quantify over x in this proposition as the introduction rule for the universal quantifier requires the variable x to have no free occurrence in the hypotheses. We cannot either eliminate the first hypothesis using the axiom $\exists f\ \forall y\ (f\ y) = x$ as the elimination rule for the existential quantifier requires the variable f to have no free occurrence in the side hypotheses, and we cannot eliminate the second hypothesis using the axiom $\forall f\ \exists g\ \forall x\ (g\ x) = f$ as the elimination rule for the existential quantifier requires the variable g to have no free occurrence in the conclusion. Thus, it seems that the proposition

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

cannot be derived from the unary comprehension scheme. We show that this is indeed the case.

Theorem 2.1 *The binary comprehension scheme is independent.*

Proof. We construct a model $\mathcal{M}$ which validates the unary comprehension scheme, but not the n-ary one. The notion of model used here is a generalization of Henkin's [7, 2], as the n-ary comprehension scheme is valid in Henkin's models.

Let T be a type, we define by induction over the structure of T the set of *tails* of T.

- if $T = \iota$ or $T = o$ then $Tails(T) = \{T\}$,
- if $T = U \to V$ then $Tails(T) = \{T\} \cup Tails(V)$.

We define, by induction over the structure of T, a family $\mathcal{M}_T$

- $\mathcal{M}_\iota$ is any set containing at least two elements,
- $\mathcal{M}_o = \{0, 1\}$,
- if $\iota \in Tails(U)$ and $U \notin Tails(T)$ then $\mathcal{M}_{T\to U}$ is the set of constant functions from $\mathcal{M}_T$ to $\mathcal{M}_U$,
- otherwise, $\mathcal{M}_{T\to U}$ is the set of all functions from $\mathcal{M}_T$ to $\mathcal{M}_U$.

An *assignment* is a function mapping every variable of type T to an element of $\mathcal{M}_T$. If φ is an assignment, we write $\varphi + (x, e)$ the assignment ψ such that $\psi(x) = e$ and $\psi(y) = \varphi(y)$ if $y \neq x$.

Let φ be an assignment, we define a function $\overline{\varphi}$, extending φ, that associates an element of $\mathcal{M}_T$ to each term of type T

- $\overline{\varphi}(x) = \varphi(x)$,
- $\overline{\varphi}(=_T) = E_T$ where E_T is a function of $\mathcal{M}_{T\to T\to o}$ such that if e and e' are elements of $\mathcal{M}_T$ then $E_T(e)(e') = 1$ if $e = e'$ and 0 otherwise,
- $\overline{\varphi}(a\ b) = \overline{\varphi}(a)(\overline{\varphi}(b))$,
- $\overline{\varphi}(\top) = 1$,
- $\overline{\varphi}(\bot) = 0$,
- $\overline{\varphi}(\neg A) = 1$ if $\overline{\varphi}(A) = 0$ and 0 otherwise,
- $\overline{\varphi}(A \wedge B) = 1$ if $\overline{\varphi}(A) = \overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \vee B) = 1$ if $\overline{\varphi}(A) = 1$ or $\overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \Rightarrow B) = 1$ if $\overline{\varphi}(A) = 0$ or $\overline{\varphi}(B) = 1$ and 0 otherwise,
- $\overline{\varphi}(A \Leftrightarrow B) = 1$ if $\overline{\varphi}(A) = \overline{\varphi}(B)$ and 0 otherwise,
- $\overline{\varphi}(\forall x\ A) = 1$ if for every e in $\mathcal{M}_T$, $\overline{\varphi + (x, e)}(A) = 1$ and 0 otherwise,
- $\overline{\varphi}(\exists x\ A) = 1$ if there exists e in $\mathcal{M}_T$, $\overline{\varphi + (x, e)}(A) = 1$ and 0 otherwise.

By an abuse of notation we write φ for $\overline{\varphi}$.

A proposition P is *valid* in $\mathcal{M}$ if $\varphi(P) = 1$ for every assignment φ.

Let T and U be two types such that $\iota \in Tails(U)$ and $U \notin Tails(T)$. Let x be a variable of type T. Let φ be an assignment and a an term of type U. We show that the function F from $\mathcal{M}_T$ to $\mathcal{M}_U$ such that for every e, $F(e) = (\varphi + (x, e))(a)$ is a constant function. By induction over the structure of a.

- If a is a variable y, then as and $U \notin Tails(T)$, $T \neq U$ and thus $y \neq x$, thus for every e, $F(e) = (\varphi + (x, e))(y) = \varphi(y)$ and F is a constant function.

- If a is a primitive symbol $=_T$, then for every e, $F(e) = E_T$ and F is a constant function.
- If $a = (b\ c)$ then let V be the type of c. The term b has the type $V \to U$. Let $G(e) = (\varphi + (x, e))(b)$ and $H(e) = (\varphi + (x, e))(c)$. As $\iota \in Tails(U)$, $\iota \in Tails(V \to U)$ and as $U \notin Tails(T)$, $V \to U \notin Tails(T)$, thus G is a constant function.
  - If $U \in Tails(V)$ then $\iota \in Tails(V)$ and $V \notin Tails(T)$. Thus, H is a constant function and $F(e) = G(e)(H(e))$ is independent of e and thus F is a constant function.
  - If $U \notin Tails(V)$ then as $\iota \in Tails(U)$, $\mathcal{M}_{V \to U}$ contains only constant functions, thus G is a constant function from $\mathcal{M}_T$ to $\mathcal{M}_{V \to U}$ and moreover the value taken by this function in $\mathcal{M}_{V \to U}$ is a constant function, thus $F(e) = G(e)(H(e))$ is independent of e and F is a constant function.
- As $\iota \in Tails(U)$, the term a does not have the form $\top$, $\bot$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$, $\forall x\ A$ or $\exists x\ A$.

Let T and U be two types, let x be a variable of type T, a be a term of type U, φ be an assignment and f be a variable of type $T \to U$ that has no occurrence in a. We show that the proposition

$$\exists f\ \forall x\ ((f\ x) = a)$$

is valid in $\mathcal{M}$.

Let F be the function from $\mathcal{M}_T$ to $\mathcal{M}_U$, $F(e) = (\varphi + (x, e))(a)$. The function F is an element of $\mathcal{M}_{T \to U}$, if $\iota \in Tails(U)$ and $U \notin Tails(T)$ because F is a constant function, otherwise because $\mathcal{M}_{T \to U}$ contains all the functions from $\mathcal{M}_T$ to $\mathcal{M}_U$.

As f has no occurrence in a, we have

$$(\varphi + (f, F) + (x, e))(f\ x) = F(e) = (\varphi + (f, F) + (x, e))(a)$$

Thus, $(\varphi + (f, F) + (x, e))((f\ x) = a) = 1$ and $\varphi(\exists f\ \forall x\ ((f\ x) = a)) = 1$.

Thus, the unary comprehension scheme is valid in this model. We check that equality axioms and extensionality axioms are valid in $\mathcal{M}$ and we show, by induction on the structure of a proof of P, that if P is a theorem of type theory with the unary comprehension scheme, then P is valid in $\mathcal{M}$.

Now, the set $\mathcal{M}_{\iota \to (\iota \to \iota)}$ contains only the constant functions from $\mathcal{M}_\iota$ to $\mathcal{M}_{\iota \to \iota}$, thus, as the set $\mathcal{M}_\iota$ contains at least two elements, the set $\mathcal{M}_{\iota \to (\iota \to \iota)}$ does not contain the function π such that for every x and y, $\pi(x)(y) = x$ and the proposition

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

is not valid in $\mathcal{M}$. Thus, it is not a theorem of type theory with the unary comprehension scheme.

Remark. The axiom of descriptions [3, 2] is also valid in the model above. The axiom of infinity [3, 2] is valid if $\mathcal{M}_\iota$ is infinite. Thus, the binary comprehension scheme is still independent if we add these axioms to type theory.

3 Extensionality

Extensionality is used several times in the proof of the equivalence between the presentations of type theory based on λ-calculus and based on the skolemized comprehension scheme. Thus, we may wonder if these theories are still equivalent if we drop the extensionality axioms. When we drop these axioms, the proposition

$$((\lambda x\ \lambda y\ \lambda z\ x)\ w\ w) = ((\lambda x\ \lambda y\ \lambda z\ y)\ w\ w)$$

is still a theorem of the λ-calculus based type theory, but we show that it is not a theorem of the hypercombinators based type theory. Similarly the proposition

$$((x, y, z \mapsto x)\ w\ w) = ((x, y, z \mapsto y)\ w\ w)$$

is not a theorem of type theory with the skolemized comprehension scheme and no extensionality axiom.

Theorem 3.1 *The proposition*

$$((x, y, z \mapsto x)\ w\ w) = ((x, y, z \mapsto y)\ w\ w)$$

is not a theorem of type theory with the skolemized comprehension scheme and no extensionality axiom.

Proof. Again, we construct a model $\mathcal{M}$ which does not validate the proposition above. The notion of model used here is a generalization of Henkin's [7, 2], as the extensionality axioms and thus the proposition above are valid in Henkin's models.

We define, by induction over the structure of T, a family $\mathcal{M}_T$

- $\mathcal{M}_\iota$ is a non empty set, we consider an element α of $\mathcal{M}_\iota$,
- $\mathcal{M}_o = \{0, 1\}$,
- $\mathcal{M}_{\iota\to\iota} = \mathcal{M}_\iota^{\mathcal{M}_\iota} - \{k_\alpha\} \cup \{K, K'\}$ where k_α is the constant function equal to α and K and K' are two objects not in $\mathcal{M}_\iota^{\mathcal{M}_\iota}$,
- $\mathcal{M}_{T\to U} = \mathcal{M}_U^{\mathcal{M}_T}$ otherwise.

An *assignment* is a function mapping every variable of type T to an element of $\mathcal{M}_T$. If φ is an assignment, we write $\varphi + (x, e)$ the assignment ψ such that $\psi(x) = e$ and $\psi(y) = \varphi(y)$ if $y \neq x$.

Let φ be an assignment, we define a function $\overline{\varphi}$, extending φ, that associates an element of $\mathcal{M}_T$ to each term of type T.

- $\overline{\varphi}(x) = \varphi(x)$.
- $\overline{\varphi}(=_T) = E_T$ where E_T is a function of $\mathcal{M}_{T\to T\to o}$ such that if e and e' are elements of $\mathcal{M}_T$ then $E_T(e)(e') = 1$ if $e = e'$ and 0 otherwise.
- $\overline{\varphi}(x_1, ..., x_n \mapsto a)$ is the function f such that $f(e_1)...(e_{n-1}) = u$ where u is defined as follows: let F be the function
$$F(c) = \overline{\varphi + (x_1, e_1) + ... + (x_{n-1}, e_{n-1}) + (x_n, c)}(a)$$
  - if F is not the constant function equal to α, we take $u = F$,
  - if F is the constant function equal to α and the term $x_1, ..., x_n \mapsto a$ is $x, y, z \mapsto x$, we take $u = K$,
  - if F is the constant function equal to α and the term $x_1, ..., x_n \mapsto a$ is not $x, y, z \mapsto x$, we take $u = K'$.
- $\overline{\varphi}(a\ b) = \overline{\varphi}(a)(\overline{\varphi}(b))$ if $\overline{\varphi}(a) \notin \{K, K'\}$ and $\overline{\varphi}(a\ b) = \alpha$ otherwise.
- $\overline{\varphi}(\top) = 1$.
- $\overline{\varphi}(\bot) = 0$.
- $\overline{\varphi}(\neg A) = 1$ if $\overline{\varphi}(A) = 0$ and 0 otherwise.
- $\overline{\varphi}(A \wedge B) = 1$ if $\overline{\varphi}(A) = \overline{\varphi}(B) = 1$ and 0 otherwise.
- $\overline{\varphi}(A \vee B) = 1$ if $\overline{\varphi}(A) = 1$ or $\overline{\varphi}(B) = 1$ and 0 otherwise.
- $\overline{\varphi}(A \Rightarrow B) = 1$ if $\overline{\varphi}(A) = 0$ or $\overline{\varphi}(B) = 1$ and 0 otherwise.
- $\overline{\varphi}(A \Leftrightarrow B) = 1$ if $\overline{\varphi}(A) = \overline{\varphi}(B)$ and 0 otherwise,
- $\overline{\varphi}(\forall x\ A) = 1$ if for every e in $\mathcal{M}_T$, $\overline{\varphi + (x, e)}(A) = 1$ and 0 otherwise.
- $\overline{\varphi}(\exists x\ A) = 1$ if there exists e in $\mathcal{M}_T$, $\overline{\varphi + (x, e)}(A) = 1$ and 0 otherwise.

By an abuse of notation we write φ for $\overline{\varphi}$.

A proposition P is *valid* in $\mathcal{M}$ if $\varphi(P) = 1$ for every assignment φ.

Let us prove that the skolemized comprehension scheme is valid in $\mathcal{M}$. The skolemized comprehension scheme is

$$\forall x_1 \ ... \ \forall x_n \ ((x_1, ..., x_n \mapsto a) \ x_1 \ ... \ x_n) = a$$

We have to prove that for every assignment φ,

$$\varphi((x_1, ..., x_n \mapsto a) \ x_1 \ ... \ x_n) = \varphi(a)$$

Let $f = \varphi(x_1, ..., x_n \mapsto a)$ and $u = \varphi((x_1, ..., x_n \mapsto a) \ x_1 \ ... \ x_{n-1}) = f(\varphi(x_1))...(\varphi(x_{n-1}))$. Let F be the function such that $F(c) = (\varphi + (x_n, c))(a)$. If F is a constant function equal to α then $u = K$ or $u = K'$ and in this case

$$\varphi((x_1, ..., x_n \mapsto a) \ x_1 \ ... \ x_n) = \alpha = \varphi(a)$$

Otherwise

$$\varphi((x_1, ..., x_n \mapsto a) \ x_1 \ ... \ x_n) = f(\varphi(x_1))...(\varphi(x_n)) = F(\varphi(x_n)) = \varphi(a)$$

We check that equality axioms are also valid in this model and thus, by induction on the structure of a proof of P, if P is a theorem of type theory without extensionality, then P is valid in $\mathcal{M}$.

Now, the term $((x, y, z \mapsto x) \ w \ w)$ denotes K and $((x, y, z \mapsto y) \ w \ w)$ denotes K', thus the proposition

$$((x, y, z \mapsto x) \ w \ w) = ((x, y, z \mapsto y) \ w \ w)$$

is not valid in $\mathcal{M}$. Thus, it is not a theorem.

Remark. The axiom of descriptions [3, 2] is also valid in the model above. The axiom of infinity [3, 2] is valid if $\mathcal{M}_\iota$ is infinite. Thus, the proposition above is still independent if we add these axioms to type theory.

Remark. The proposition above is formulated in type theory with the skolemized comprehension scheme. The question of the existence of a proposition in the language of the (non skolemized) comprehension scheme, i.e. without abstractions that would be provable in the presentation with λ-calculus and not provable in the presentation with the comprehension scheme is left open. The natural candidate given by the translation of definition 1.10

$$\exists f \ \exists g \ (\forall x \ \forall y \ \forall z \ ((f \ x \ y \ z) = x) \ \wedge \forall x \ \forall y \ \forall z \ ((g \ x \ y \ z) = y) \wedge \forall w \ ((f \ w \ w) = (g \ w \ w)))$$

is unfortunately provable in type theory with the comprehension scheme. (Notice that the proposition 1.12 fails when extensionality is dropped.) Indeed, consider f such that $\forall x \ \forall y \ \forall z \ (f \ x \ y \ z) = x$, C such that $\forall u \ \forall x \ \forall y \ (C \ u \ x \ y) = (u \ y \ x)$ and $g = (C \ f)$. We have $(f \ x \ y \ z) = x$, $(g \ x \ y \ z) = (C \ f \ x \ y \ z) = (f \ y \ x \ z) = y$ and $(g \ w \ w) = (C \ f \ w \ w) = (f \ w \ w)$.

Remark. In [8] Henkin presents a proof of the equivalence of a presentation of second order logic based on a rule of substitution of functional variables and a presentation based on a comprehension scheme. As, in the second order logic presented in this paper, there seems to be no equality between functions or predicates, the proof goes through without the extensionality axioms. It seems that the extensionality axioms are required to translate the full type theory.

4 Skolemizing the open comprehension scheme

Type theory can be presented either with the closed comprehension scheme, or the open one. When we skolemize the closed comprehension scheme we get hyper-combinators. In this section we characterize the language obtained by skolemizing the open comprehension scheme.

When we skolemize an instance of the open comprehension scheme

$$\forall x_1 \ ... \ \forall x_n \ \exists f \ \forall y_1 \ ... \ \forall y_p \ (f \ y_1 \ ... \ y_p) = a$$

then we introduce a function symbol $f_{(x_1,...,x_n),(y_1,...,y_p),a}$ and the axiom

$$\forall x_1 \ ... \ \forall x_n \ \forall y_1 \ ... \ \forall y_p \ ((f_{(x_1,...,x_n),(y_1,...,y_p),a} \ x_1 \ ... \ x_n) \ y_1 \ ... \ y_p) = a$$

Remark. In contrast with the skolemization of the closed comprehension scheme (section 1.4), in this case the existential quantifier is not external. Thus skolemization must be used with care. If we let the symbol $f_{(x_1,...,x_n),(y_1,...,y_p),a}$ be a primitive symbol, then the skolemization rule is unsound unless the axiom of choice is assumed in the theory. As remarked by Miller [10, 11], sound skolemization requires the symbol $f_{(x_1,...,x_n),(y_1,...,y_p),a}$ to be a function symbol, i.e. the symbol $f_{(x_1,...,x_n),(y_1,...,y_p),a}$ alone is not a term, but if $b_1, ..., b_n$ are terms then $(f_{(x_1,...,x_n),(y_1,...,y_p),a} \ b_1 \ ... \ b_n)$ is a term.

Proposition 4.1 *Let a be a term containing no Skolem symbol. Let $x_1, ..., x_n$, $y_1, ..., y_p$ be variables such that all the free variables of a are among $x_1, ..., x_n$, $y_1, ..., y_p$. Let $b_1, ..., b_n$ be terms containing no Skolem symbol. Let $y'_1, ..., y'_p$ be variables not occurring free in these terms and let $x'_1, ..., x'_{n'}$ be variables such that the free variables of $a[y_1 \leftarrow y'_1]...[y_p \leftarrow y'_p][x_1 \leftarrow b_1]...[x_n \leftarrow b_n]$ are among $x'_1, ..., x'_{n'}, y'_1, ..., y'_p$. We have*

$$(f_{(x_1,...,x_n),(y_1,...,y_p),a} \ b_1 \ ... \ b_n) = (f_{(x'_1,...,x'_{n'}),(y'_1,...,y'_p),a[y_1\leftarrow y'_1]...[y_p\leftarrow y'_p][x_1\leftarrow b_1]...[x_n\leftarrow b_n]} \ x'_1 \ ... \ x'_{n'})$$

Proof. We have

$$\forall x_1 \ ... \ \forall x_n \ \forall y_1 \ ... \ \forall y_p \ ((f_{(x_1,...,x_n),(y_1,...,y_p),a} \ x_1 \ ... \ x_n \ y_1 \ ... \ y_p) = a)$$

thus

$$\forall x_1 \ ... \ \forall x_n \ \forall y'_1 \ ... \ \forall y'_p \ ((f_{(x_1,...,x_n),(y_1,...,y_p),a} \ x_1 \ ... \ x_n \ y'_1 \ ... \ y'_p) = a[y_1 \leftarrow y'_1, ..., y_p \leftarrow y'_p])$$

$$(f_{(x_1,...,x_n),(y_1,...,y_p),a} \ b_1 \ ... \ b_n \ y'_1 \ ... \ y'_p) = a[y_1 \leftarrow y'_1]...[y_p \leftarrow y'_p][x_1 \leftarrow b_1]...[x_n \leftarrow b_n]$$

We have

$$\forall x'_1 \ ... \ \forall x'_{n'} \ \forall y'_1 \ ... \ \forall y'_p \ (f_{(x'_1,...,x'_{n'}),(y'_1,...,y'_p),a[y_1\leftarrow y'_1]...[y_p\leftarrow y'_p][x_1\leftarrow b_1]...[x_n\leftarrow b_n]} \ x'_1 \ ... \ x'_{n'} \ y'_1 \ ... \ y'_p) = a[y_1 \leftarrow y'_1]...[y_p \leftarrow y'_p][x_1 \leftarrow b_1]...[x_n \leftarrow b_n]$$

thus

$$(f_{(x_1,...,x_n),(y_1,...,y_p),a} \ b_1 \ ... \ b_n \ y'_1 \ ... \ y'_p) = ((f_{(x'_1,...,x'_{n'}),(y'_1 \ ... \ y'_p),a[y_1\leftarrow y'_1]...[y_p\leftarrow y'_p][x_1\leftarrow b_1]...[x_n\leftarrow b_n]} \ x'_1 \ ... \ x'_{n'} \ y'_1 \ ...y'_p))$$

$$\forall y'_1 \ ... \ \forall y'_p \ (f_{(x_1,...,x_n),(y_1,...,y_p),a} \ b_1 \ ... \ b_n \ y'_1 \ ... \ y'_p) = ((f_{(x'_1,...,x'_{n'}),(y'_1 \ ... \ y'_p),a[y_1\leftarrow y'_1]...[y_p\leftarrow y'_p][x_1\leftarrow b_1]...[x_n\leftarrow b_n]} \ x'_1 \ ... \ x'_{n'} \ y'_1 \ ...y'_p))$$

and by extensionality

$$(f_{(x_1,...,x_n),(y_1,...,y_p),a} \ b_1 \ ... \ b_n) = (f_{(x'_1,...,x'_{n'}),(y'_1,...,y'_p),a[y_1\leftarrow y'_1]...[y_p\leftarrow y'_p][x_1\leftarrow b_1]...[x_n\leftarrow b_n]} \ x'_1 \ ... \ x'_{n'})$$

Remark. We consider a restriction of the language above, in such a way that a symbol $f_{(x_1,...,x_n),(y_1,...,y_p),a}$ can only be applied to the variables $x_1, ..., x_n$. We write $y_1, ..., y_p \mapsto a$ for the term $(f_{(x_1,...,x_n),(y_1,...,y_p),a} \ x_1 \ ... \ x_n)$ where a is a term containing no Skolem symbol.

Notice that the free variables of $y_1, ..., y_p \mapsto a$ are $x_1, ..., x_n$, i.e. the free variables of a but $y_1, ..., y_p$. As a corollary of the previous proposition we have

$$(y_1, ..., y_p \mapsto a)[x \leftarrow b] = (y'_1, ..., y'_p \mapsto a[y_1 \leftarrow y'_1]...[y_p \leftarrow y'_p][x \leftarrow b])$$

if a and b are terms containing no Skolem symbol, and $y'_1, ..., y'_n$ variable not occurring in these terms.

This language is larger than the language of hyper-combinators since it allows free variables in the body of abstractions and substitution with renaming, but it is not λ-calculus, since it does not allow nested abstractions.

Remark. If $a, b_1, ..., b_n$ are terms without Skolem symbols we have

$$(f_{(x_1,...,x_n),(y_1,...,y_p),a}\ b_1\ ...\ b_n) = y'_1, ..., y'_p \mapsto a[y_1 \leftarrow y'_1]...[y_p \leftarrow y'_p][x_1 \leftarrow b_1]...[x_n \leftarrow b_n]$$

We can extend the notation above and write

$$y'_1, ..., y'_p \mapsto a[y_1 \leftarrow y'_1]...[y_p \leftarrow y'_p][x_1 \leftarrow b_1]...[x_n \leftarrow b_n]$$

for the term $(f_{(x_1,...,x_n),(y_1,...,y_p),a}\ b_1\ ...\ b_n)$ even if the terms $b_1, ..., b_n$ contain Skolem symbols.

This language allows free variables in the body of abstractions, substitution under abstractions with renaming and nested abstractions, but it is not λ-calculus because a variable bound in an abstraction cannot be bound upper in the term in another abstraction. For instance the term $x \mapsto (y \mapsto x)$ cannot be constructed in this language.

Remark. If we skolemize this way the unary comprehension scheme, we get a language with unary functions and no variable binding through abstractions. In this language there is no term for the first projection since we cannot prove the proposition $\exists f\ \forall x\ \forall y\ (f\ x\ y) = x$. Thus to define this function, we need either variable binding through abstractions $x \mapsto (y \mapsto x)$ or n-ary functions $x, y \mapsto x$.

5 Set theory

Like type theory, set theory can be presented either with existence axioms (Zermelo's axioms or refinements), with an explicit language for objects obtained by skolemizing these axioms or with a language with a binder, including symbols $\mathcal{P}$, $\bigcup$ and $\{,\}$ where $\mathcal{P}(A)$ is the power set of A, $\bigcup(A)$ is the union of the elements of A and $\{,\}(A, B)$ is the pair containing A and B and a notation $\{x \in A \mid P\}$ for the subset of A of the elements verifying P. In this section, we show the equivalence of these presentations.

5.1 Set theory with a binder

Definition 5.1 *Terms and Propositions* are inductively defined by

- variables are terms,
- if a is a term then $\mathcal{P}(a)$ is a term,
- if a is a term then $\bigcup(a)$ is a term,
- if a and b are terms then $\{,\}(a, b)$ (i.e. $\{a, b\}$) is a term,
- if A is a term and P a proposition then $\{z \in A \mid P\}$ is a term,
- if a and b are terms then $a \in b$ is a proposition,
- if a and b are terms then $a = b$ is a proposition,
- $\top$ and $\bot$ are propositions,
- if A is 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 proposition and x a variable then $\forall x\ A$ and $\exists x\ A$ are propositions.

Substitution is defined as in λ-calculus with renaming to avoid captures in the terms $\{z \in A \mid P\}$.

Definition 5.2 (Axioms)
Conversion (power set, union, pair and subset scheme):

$$\forall x\ \forall y\ ((y \in \mathcal{P}(x)) \Leftrightarrow \forall z\ ((z \in y) \Rightarrow (z \in x)))$$

$$\forall x\ \forall y\ ((y \in \bigcup(x)) \Leftrightarrow \exists z\ ((y \in z) \wedge (z \in x)))$$

$$\forall x\ \forall y\ \forall z\ ((z \in \{,\}(x,y)) \Leftrightarrow ((z = x) \vee (z = y)))$$

$$\forall x_1\ ...\ \forall x_n\ \forall y\ \forall z\ ((z \in \{z \in y \mid P\}) \Leftrightarrow ((z \in y) \wedge P))$$

Extensionality:

$$\forall x\ \forall y\ ((\forall z\ ((z \in x) \Leftrightarrow (z \in y))) \Rightarrow (x = y))$$

Equality:

$$\forall x\ (x = x)$$

$$\forall w_1\ ...\ \forall w_p\ \forall x\ \forall y\ ((x = y) \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y]))$$

Deduction rules are the usual ones.

5.2 Set theory with existence axioms

Definition 5.3 (Zermelo's set theory)
Comprehension (power set, union, pair and subset scheme):

$$\forall x\ \exists A\ \forall y\ ((y \in A) \Leftrightarrow \forall z\ ((z \in y) \Rightarrow (z \in x)))$$

$$\forall x\ \exists A\ \forall y\ ((y \in A) \Leftrightarrow \exists z\ ((y \in z) \wedge (z \in x)))$$

$$\forall x\ \forall y\ \exists A\ \forall z\ ((z \in A) \Leftrightarrow ((z = x) \vee (z = y)))$$

$$\forall x_1\ ...\ \forall x_n\ \forall y\ \exists A\ \forall z\ ((z \in A) \Leftrightarrow ((z \in y) \wedge P))$$

where A is not free in P and $x_1, ..., x_n$ are the free variables of P different from z.
Extensionality:

$$\forall x\ \forall y\ ((\forall z\ ((z \in x) \Leftrightarrow (z \in y))) \Rightarrow (x = y))$$

Equality:

$$\forall x\ (x = x)$$

$$\forall w_1\ ...\ \forall w_p\ \forall x\ \forall y\ ((x = y) \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y]))$$

5.3 Set theory with skolemized axioms

The skolemization of the power set axiom, the union axiom and the pair axiom introduces function symbols $\mathcal{P}$, $\bigcup$ and $\{,\}$. When we skolemize the subset axiom, we introduce, for each proposition P that does not contain Skolem symbols and sequence of symbols $x_1, ..., x_n, z$ such that all the free variables of P are among $x_1, ..., x_n, z$ a function symbol $f_{x_1,...,x_n,z,P}$ and an axiom

$$\forall x_1\ ...\ \forall x_n\ \forall y\ \forall z\ (z \in (f_{x_1,...,x_n,z,P}\ x_1\ ...\ x_n\ y) \Leftrightarrow ((z \in y) \wedge P))$$

By Skolem's theorem, the theory obtained this way is a conservative extension of set theory.

Now, we want to prove a result similar to proposition 1.12 and corollary 1.13, i.e. for every proposition P, we want to build a Skolem symbol free proposition P° that is provable, if and only if P is provable in the skolemized theory. In fact, we shall prove such a result for an arbitrary first order theory and then apply this result to set theory.

We consider a theory with axioms of the form $\forall x_1\ ...\ \forall x_n\ \exists y\ P$ and we skolemize them as $\forall x_1\ ...\ \forall x_n\ P[y \leftarrow (f\ x_1\ ...\ x_n)]$.

Definition 5.4 Let A be a proposition, we define the proposition A° as follows

- If A is atomic, we replace every subterm of the form $(f_i\ t_{i,1}\ ...\ t_{i,n_i})$ by a variable y_i we get a proposition A_1, let

  $A^\circ = \exists y_1\ ...\ \exists y_p$

  $$((P_1[x_1 \leftarrow t_{1,1}, ..., x_{n_1} \leftarrow t_{1,n_1}, y \leftarrow y_1])^\circ \wedge ... \wedge (P_p[x_1 \leftarrow t_{p,1}, ..., x_{n_p} \leftarrow t_{p,n_p}, y \leftarrow y_p])^\circ \wedge A_1)$$

- $\top^\circ = \top$, $\bot^\circ = \bot$, $(\neg A)^\circ = \neg A^\circ$, $(A \wedge B)^\circ = A^\circ \wedge B^\circ$, $(A \vee B)^\circ = A^\circ \vee B^\circ$, $(A \Rightarrow B)^\circ = A^\circ \Rightarrow B^\circ$, $(A \Leftrightarrow B)^\circ = A^\circ \Leftrightarrow B^\circ$,
- $(\forall x\ A)^\circ = \forall x\ A^\circ$, $(\exists x\ A)^\circ = \exists x\ A^\circ$.

Consider a proposition A, call n the multiset of the sizes of the subterms of A that whose head is a Skolem symbol, call p the number of occurrences of connectors and quantifiers in A. This definition is by induction over $< n, p >$.

Proposition 5.1 *If for every axiom of the form*

$$\forall x_1\ ...\ \forall x_n\ \exists y\ P$$

we can prove the following unicity property

$$\forall x_1\ ...\ \forall x_n\ \forall u\ \forall v\ ((P[y \leftarrow u] \wedge P[y \leftarrow v]) \Rightarrow (u = v))$$

Then for every proposition A, the proposition $A \Leftrightarrow A^\circ$ is provable in the skolemized theory.

Proof. By induction over $< n, p >$. Take a proposition A, assume the property for every proposition lower than A for the order above.

- If A is an atomic proposition, assume A, we prove A° by giving the terms $(f_i\ t_{i,1}\ ...\ t_{i,n_i})$ for y_i. Conversely, assume A°, from the hypotheses

  $$(P_1[x_1 \leftarrow t_{1,1}, ..., x_{n_1} \leftarrow t_{1,n_1}, y \leftarrow y_1])^\circ$$

  $$...$$

  $$(P_p[x_1 \leftarrow t_{p,1}, ..., x_{n_p} \leftarrow t_{p,n_p}, y \leftarrow y_p])^\circ$$

  and the induction hypothesis we get

  $$P_1[x_1 \leftarrow t_{1,1}, ..., x_{n_1} \leftarrow t_{1,n_1}, y \leftarrow y_1]$$

  $$...$$

  $$P_p[x_1 \leftarrow t_{p,1}, ..., x_{n_p} \leftarrow t_{p,n_p}\ y \leftarrow y_p]$$

  Then with the unicity property we get

  $$y_1 = (f_1\ t_{1,1}\ ...\ t_{1,n_1})$$

  $$...$$

  $$y_p = (f_p\ t_{p,1}\ ...\ t_{p,n_p})$$

  and with the equality axioms and the hypothesis A_1 we get A. Then using the elimination rule of the existential quantifier we get a proof of A from A°
- If A is not atomic we apply the induction hypothesis.

Corollary 5.2 *The proposition A is provable in the skolemized theory if and only if A° is provable in the non skolemized one.*

Corollary 5.3 *For every proposition A in the language of the skolemized set theory, we can build a proposition A° in the language of the (non skolemized) set theory such that A is provable in the skolemized set theory if and only if A° is provable in the (non skolemized) set theory.*

For instance if $A = (y \in \bigcup(x))$ then

$$A^\circ = \exists z\ (\forall u\ ((u \in z) \Leftrightarrow \exists v\ ((u \in v) \wedge (v \in x))) \wedge (y \in z))$$

5.4 Set theory with a binder and the skolemized axioms

When we interpret the term $(f_{x_1,...,x_n,z,P}\ a_1\ ...\ a_n\ A)$ as the term $\{z \in A \mid P[x_1 \leftarrow a_1, ..., x_n \leftarrow a_n]\}$, the language of the skolemized set theory can be seen as a sub-language of that of set theory with a binder.

We define a translation of set theory with a binder to the language of the skolemized theory. This transformation is analogous to λ-lifting.

Definition 5.5

- $x' = x$,
- $\{z \in A \mid P\}' = \{x \in A' \mid P'^\circ\} = (f_{x_1,...,x_n,z,P'^\circ}\ x_1\ ...\ x_n\ A')$, where $x_1, ..., x_n$ are the variables free in P'° and different from z,
- $(\bigcup(a))' = \bigcup(a')$, $(\mathcal{P}(a))' = \mathcal{P}(a')$, $\{a, b\}' = \{a', b'\}$,
- $(a = b)' = (a' = b')$, $(a \in b)' = (a' \in b')$,
- $\top' = \top$, $\bot' = \bot$,
- $(\neg A)' = \neg A'$,
- $(A \wedge B)' = (A' \wedge B')$, $(A \vee B)' = (A' \vee B')$, $(A \Rightarrow B)' = (A' \Rightarrow B')$, $(A \Leftrightarrow B)' = (A' \Leftrightarrow B')$,
- $(\forall x\ A)' = \forall x\ A'$, $(\exists x\ A)' = \exists x\ A'$.

Proposition 5.4 *If a is a term, the proposition $a = a'$ is provable set theory with a binder. If a is a proposition, the proposition $a \Leftrightarrow a'$ is provable set theory with a binder.*

Proof. By induction on the structure of a.

- If $a = \{z \in A \mid P\}$ then by extensionality we are reduced to prove

  $$(z \in a) \Leftrightarrow (z \in a')$$

  i.e.

  $$((z \in A) \wedge P) \Leftrightarrow ((z \in A') \wedge P'^\circ)$$

  This is a consequence of the induction hypotheses $A' = A$, $P \Leftrightarrow P'$ and the proposition $P' \Leftrightarrow P'^\circ$.
- The other cases are application of the induction hypothesis.

Proposition 5.5 *If A is an axiom of set theory with a binder then A' is provable in the skolemized set theory.*

Proof.

- If A is an instance of the scheme

$$\forall x_1 \ ... \ \forall x_n \ \forall y \ \forall z \ ((z \in \{z \in y \mid P\}) \Leftrightarrow ((z \in y) \wedge P))$$

then we have to prove

$$\forall x_1 \ ... \ \forall x_n \ \forall y \ \forall z \ ((z \in (f_{x_1,...,x_n,z,P'^\circ} \ x_1 \ ... \ x_n \ y)) \Leftrightarrow ((z \in y) \wedge P'))$$

As an axiom of the skolemized theory we have

$$z \in (f_{x_1,...,x_n,z,P'^\circ} \ x_1 \ ... \ x_n \ y) \Leftrightarrow (z \in y) \wedge P'^\circ$$

and we conclude with $P' \Leftrightarrow P'^\circ$.

- If A has the form

$$\forall w_1 \ ... \ \forall w_p \ \forall x \ \forall y \ (x = y) \Rightarrow (P[z \leftarrow x] \Rightarrow P[z \leftarrow y])$$

then A' is

$$\forall w_1 \ ... \ \forall w_p \ \forall x \ \forall y \ (x = y) \Rightarrow (P'[z \leftarrow x] \Rightarrow P'[z \leftarrow y])$$

which is an axiom.

- Otherwise the axiom is its own translation.

Proposition 5.6 *If a and b are terms of set theory with a binder then the proposition*

$$(a[x \leftarrow b])' = a'[x \leftarrow b']$$

is provable in the skolemized set theory.

If P is a proposition of the set theory with a binder and b is a term of the set theory with a binder then the proposition $(P[x \leftarrow a])' \Leftrightarrow P'[x \leftarrow a']$ is provable in the skolemized set theory.

Proof. By induction over the structure of a and P.

- If $a = \{z \in A \mid P\}$, where z is a fresh variable.
  - If x is not free in P then let $y_1, ..., y_n$ be the free variables of P different from z. We have

$$a'[x \leftarrow b'] = (f_{y_1,...,y_n,z,P'^\circ} \ y_1 \ ... \ y_n \ A'[x \leftarrow b'])$$

$$(a[x \leftarrow b])' = (f_{y_1,...,y_n,z,P'^\circ} \ y_1 \ ... \ y_n \ (A[x \leftarrow b])')$$

By extensionality we are reduced to prove that

$$(z \in a'[x \leftarrow b']) \Leftrightarrow (z \in (a[x \leftarrow b])')$$

i.e.

$$((z \in A'[x \leftarrow b']) \wedge P'^\circ) \Leftrightarrow ((z \in (A[x \leftarrow b])') \wedge P'^\circ)$$

this follows from the induction hypothesis $A'[x \leftarrow b'] = (A[x \leftarrow b])'$.

– Otherwise let $x, y_1, ..., y_n$ be the free variable of P different from z. We have

$$a'[x \leftarrow b'] = (f_{x,y_1,...,y_n,z,P'^\circ}\ b'\ y_1\ ...\ y_n\ A'[x \leftarrow b']\ )$$

$$(a[x \leftarrow b])' = (f_{u_1,...,u_p,y_1,...,y_n,z,(P[x\leftarrow b])'^\circ}\ u_1\ ...\ u_p\ y_1\ ...\ y_n\ (A[x \leftarrow b])')$$

By extensionality we are reduced to prove that

$$(z \in a'[x \leftarrow b']) \Leftrightarrow (z \in (a[x \leftarrow b])')$$

i.e.

$$((z \in A'[x \leftarrow b']) \wedge (P'^\circ[x \leftarrow b'])) \Leftrightarrow ((z \in (A[x \leftarrow b])') \wedge (P[x \leftarrow b])'^\circ)$$

this follows from $P' \Leftrightarrow P'^\circ$, $(P[x \leftarrow b])' \Leftrightarrow (P[x \leftarrow b])'^\circ$ and the induction hypotheses $A'[x \leftarrow b'] = (A[x \leftarrow b])'$ and $P'[x \leftarrow b'] \Leftrightarrow (P[x \leftarrow b])'$.

- The other cases are application of the induction hypothesis.

Proposition 5.7 *The proposition P is provable in set theory with a binder if and only if P' is provable in the skolemized set theory.*

Proof. Assume P is provable in set theory with a binder. By induction on the structure of the proof of P we show that P' is provable in the skolemized set theory. Most cases are trivial, we use the the proposition 5.5 for the axiom rule and the proposition 5.6 for the elimination rule of the universal quantifier and the introduction rule of the existential quantifier.

Conversely, if P' is provable in the skolemized set theory then P' is provable in set theory with a binder. As the proposition $P \Leftrightarrow P'$ is provable set theory, then so is P.

Remark. The translation from the theory with a binder to the skolemized theory is simpler in set theory than in type theory (where it requires the n-ary conversion scheme). This is because P is a proposition in $\{x \in A \mid P\}$ while a is a term in $\lambda x\ a$. Thus in set theory, we can use the fact that every proposition of the skolemized language is equivalent to one without Skolem symbols, while no such results holds for terms in type theory.

Proposition 5.8 *Set theory with a binder is a conservative extension of the skolemized set theory.*

Proof. If P is a proposition in the language of the skolemized set theory, then $P' = P$. Thus P is provable in set theory with a binder if and only if it is provable in the skolemized theory.

Proposition 5.9 *The set theory with a binder and the skolemized set theory are equivalent both in the sense that the former is a conservative extension of the latter and that it can be translated into it preserving provability.*

Theorem 5.1 *The three presentations of set theory (based on existence axioms, on skolemized existence axioms and with a binder) are equivalent, both in the sense that each one is a conservative extension of the previous and that each one can be encoded into the previous preserving provability.*

Acknowledgements

The author thanks Thérèse Hardin and Gérard Huet for their help in the preparation of this paper and the anonymous referees for many helpful remarks.

References

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

[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] H. Curry, An analysis of logical substitution, *American Journal of Mathematics*, 51 (1929), pp. 363-384.

[5] G. Dowek, Collections, types and sets, *manuscript* (1994).

[6] J.Y. Girard, Y. Lafont, P. Taylor, Proofs and Types, *Cambridge University Press* (1989).

[7] L. Henkin, Completeness in the theory of types, *The Journal of Symbolic Logic*, 15 (1950) pp. 81-91.

[8] L. Henkin, Banishing the rule of substitution for functional variables, *The Journal of Symbolic Logic*, 18, 3 (1953), pp. 201-208.

[9] R.J.M. Hughes, Super-combinators, a new implementation method for applicative languages, *Proceedings of Lisp and Functional Programming* (1982).

[10] D.A. Miller, Proofs in higher order logic, *PhD Thesis*, Carnegie Mellon University (1983).

[11] D.A. Miller, A compact representation of proofs, *Studia Logica*, 46, 4 (1987).

[12] L.C. Paulson, Set theory for verification: I. From foundations to functions. *Journal of Automated Reasoning* 11 (1993) pp. 353-389.

[13] T. Skolem, Über die mathematische logik, *Norsk Matematisk Tidsskrift*, 10 (1928), pp. 125-142.

Unité de recherche Inria Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 Villers Lès Nancy
Unité de recherche Inria Rennes, Irisa, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Unité de recherche Inria Rhône-Alpes, 46 avenue Félix Viallet, 38031 Grenoble Cedex 1
Unité de recherche Inria Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex
Unité 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)
ISSN 0249-6399